UAH

Construcción de mapas probabilísticos mediante técnicas de SLAM en entorno ROS

Máster Universitario en Sistemas electronicos avanzados. Sistemas Inteligentes. Departamento de Electrónica

Presentado por:
D. Daniel Julián Aguilar

Dirigido por:
Dra. María Elena López Guillén
Dr. Fernando Herranz Cabrilla

Alcalá de Henares, a 18 de Septiembre de 2014

"Nunca consideres el estudio como una obligación,
sino como una oportunidad para penetrar en
el bello y maravilloso mundo del saber"
Albert Einstein

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Resumen

Resumen

El presente trabajo es una mejora del proyecto fin de carrera con título "Construcción de Mapas Probabilísticos mediante sensores de visión y láser a bordo de un robot" [1], donde se soluciona el problema del SLAM para entornos pequeños usando un robot dotado de un par de cámaras a modo de sistema de visión estéreo y con la información de odometría disponible. El enfoque seguido en el trabajo que se usa como referencia se plantea como un filtro de Kalman Extendido (EKF), que mantiene una estimación de la posición del robot y las marcas naturales que definen el entorno.

Tras detectar los puntos débiles encontrados en el proyecto mencionado en cuanto al límite del tamaño del mapa, se ha procedido a cambiar el filtro de Kalman extendido que se usa como método de inferencia bayesiano para la gestión del mapa por un enfoque más actual de *Smoothing and Mapping*. Tras la implementación de la solución al SLAM con el método de *Smoothing and Mapping* se han comparado los resultados obtenidos con los que se obtuvieron haciendo uso del EKF. Se ha adaptado y optimizado el método de asociación de datos ya utilizado y que tan buenos resultados dio.

Para esto se ha realizado una versión del algoritmo iSAM [2] con un enfoque $multi-proceso$, permitiendo mezclar las bondades del trabajo realizado en [1] con la mejora de un enfoque más moderno a la solución del SLAM.

El algoritmo se ha desarrollado haciendo uso de las librerías GTSAM. Estas librerías ayudan a implementar el *Smoothing and Mapping* ofreciendo clases para manejar redes bayesianas. La propuesta funciona como varios nodos del meta-sistema operativo ROS (Robot Operating System) que se comunican entre sí y funcionan en paralelo.

El algoritmo propuesto se ha desarrollado de manera que sea compatible con diferentes plataformas robóticas y diferentes sensores de visión estereoscópica.

La plataforma robótica utilizada es Qbo, desarrollado por la empresa TheCorpora, sobre la que se han validado los resultados de los dos enfoques seguidos, demostrando que el efoque más actual de *Smoothing and Mapping* supera a la solución clásica de SLAM EKF tanto en el tamaño del mapa que se puede gestionar como en el tiempo de cómputo de cada iteración del algoritmo.

Abstract

This project is an new version of the final year project called "Construcción de Mapas Probabilísticos mediante sensores de visión y láser a bordo de un robot" [1], in which the SLAM problem is solved for small environments using a robot provided with two cameras that compoese the stereoscopic vision system and also with the odometry available. The solution given in the project cited is an Extended Kalman Filter (EKF) that mantains an estimate of the position and the natural landmarks that model the environment.

There were some weak points found in the original project, such as a limited map size. In order to improve its behaviour, the EKF used as a bayesian inference method to manage the map has been changed. A Smoothing and Mapping method has been used in its place. The results obtained with the two methods have been compared. The data association method is the same in the two cases with just a few modifications to adapt it to the inputs necesary for the Smoothing and Mapping.

The present work is a version of the iSAM [2] algorithm with a $multi - process$ implementation, allowing to mix the strong points of the solution provided in [1] with an improvement coming from a modern approach to the SLAM problem.

The GTSAM libraries have been used to develop the algorithm. These libraries help to implement the Smoothing and Mapping by the use of classes to manage bayesian networks. The final solution runs as various nodes of the meta-operative system ROS (Robot Operating System) who communicate between them and work in paralell.

The algorithm proposed has been developed to be adaptable to different robotic platforms and also different stereoscopic vision sensors.

The robotic platform used is Qbo, from TheCorpora company. It has been used to validate the results of the two tested solutions, concluding that the more actual method of Smoothing and Mapping improves the clasic SLAM EKF both in the map size and the computation time spent in one step of the algorithm.

Capítulo 2

Introducción

2.1. Estado del Arte

2.1.1. Historia del problema SLAM

El problema de la localización y mapeo simultáneo (SLAM) trata de resolver si es posible que un robot móvil esté localizado en un entorno desconocido y a la vez construir un mapa consistente de dicho entorno. El mapa obtenido depende de la tecnología sensorial incluida en el robot. Así, en el caso de utilizar sensores de barrido láser o ultrasónico, el mapa que se obtiene suele limitarse a un plano de altura fija y se compone de rectas o puntos de interés. En el caso de cámaras el mapa que se obtienen suele representarse como un conjunto disperso de puntos tridimensionales del entorno. La construcción del mapa acorde a los sensores utilizados es una tarea muy importante en los sistemas SLAM. Se suele considerar a la solución al problema SLAM como el "santo grial" para la comunidad que trabaja entorno a la robótica móvil, ya que ofrece las herramientas necesarias para realizar un robot verdaderamente autónomo.

El problema SLAM se ha formulado y resuelto de manera teórica en numerosas ocasiones. También se han implementado soluciones para diferentes tipos de entorno, como interiores, exteriores, ambientes subacuáticos y espacios aéreos. SLAM se puede considerar un problema resuelto a nivel teórico y conceptual. Sin embargo, problemas importantes a la hora de poner en práctica los algoritmos teóricos hacen que se continúen investigando más soluciones del SLAM. Estas soluciones se están enfocando sobretodo en la creación y uso de mapas perceptualmente más ricos como parte del algoritmo SLAM.

Se comenzó a hablar del problema del SLAM probabilístico en el año 1986 durante la IEEE Robotics and Automation Conference, celebrada en San Francisco, California. En esa época los métodos probabilísticos acababan de empezar a introducirse tanto en la robótica como en la inteligencia artificial (AI). Algunos investigadores habían buscado aplicar métodos de estimación teórica para problemas de mapeo y localización; entre ellos se incluían Peter Cheeseman, Jim Crowley y Hugh Durrant-Whyte. En el transcurso de la conferencia se debatió mucho sobre la creación de mapas consistentes. Durante ésta, Raja Chatila, Oliver Faugeras, Randal Smith y más investigadores contribuyeron a la conversación.

El resultado de esa conversación fue el reconocimiento de que la creación de mapas probabilísticos consistentes era un problema fundamental en la robótica con importantes cuestiones tanto conceptuales como computacionales por ser abordadas. A lo largo de los siguientes años se publicaron algunos de los trabajos clave de los temas tratados en la conferencia. Los trabajos hechos por Smith y Cheesman [3] y Durrant-Whyte [4] establecieron las bases para describir relaciones entre marcas del entorno y para manipular la incertidumbre geométrica. Un elemento clave de estos trabajos fue el mostrar que existe un alto grado de correlación entre las estimaciones de la localización de diferentes marcas del entorno en un mapa y que, en efecto, estas correlaciones crecerían con sucesivas observaciones.

Al mismo tiempo Ayache y Faugeras [5] estaban llevando a cabo los primeros trabajos en navegación visual, Crowley [6] y Chatila y Laumond [7] estaban trabajando en navegación de robots móviles basada en sonar, usando algoritmos basados en el filtro de Kalman. Estas dos vertientes de la investigación tenían mucho en común y convergieron poco después en el estudio sobre marcas en el entorno de Smith et al. [8]. Este estudio mostró que según un robot se mueve por un entorno desconocido realizando observaciones relativas de marcas, las estimaciones de esas marcas están todas ellas correlacionadas entre sí porque comparten el error común de la estimación de la localización del propio robot [9]. Las implicaciones de esta afirmación fueron importantes: Una solución consistente al problema de la combinación del mapeo y la localización simultáneos requeriría un estado conjunto del sistema compuesto por la posición del robot y la

posición de cada una de las marcas del entorno, que debía ser actualizado con cada observación de una marca. A su vez, esto requería que el estimador utilizara un gran vector de estados (del orden del número de marcas mantenidas en el mapa) con un coste computacional del cuadrado del número de marcas.

Este trabajo no se centró en las propiedades de convergencia del mapa o en su estado estacionario. De hecho, se asumió al momento que los errores en la estimación del mapa no convergerían y que de hecho mostrarían un comportamiento "random-walk" con un crecimiento sin límites del error. Por lo tanto, dada la complejidad computacional del problema de la realización del mapa y sin en conocimiento del comportamiento convergente del mapa, los investigadores se centraron en una serie de aproximaciones al problema de la creación consistente de un mapa, que asumían o incluso forzaban la minimización o incluso la eliminación de las correlaciones entre marcas de manera que se redujera el filtro completo a una serie de filtros vehículo-marca desacoplados ([10] y [11] por ejemplo). Por la misma razón, el trabajo teórico sobre la combinación de la localización y el mapeo estuvo temporalmente detenido, con los esfuerzos centrados tanto en el mapeo como en la localización como problemas separados.

El avance conceptual vino con la realización de que el problema combinado de mapeo y localización, una vez formulado como un problema de estimación única, era en realidad convergente. Más importante aún, se reconoció que las correlaciones entre los puntos de referencia, que la mayoría de los investigadores habían tratado de reducir al mínimo, eran en realidad la parte más crítica del problema y que, al contrario de lo que se pensaba, cuanto más crezca la correlación mejor será la solución. La estructura del problema SLAM, los resultados sobre la convergencia y la acuñación de las siglas SLAM fue presentado por primera vez en un trabajo sobre la robótica móvil en el Simposio Internacional de Investigación Robótica de 1995 [12]. La teoría fundamental sobre la convergencia y muchos de los resultados iniciales fueron desarrolladas por Csorba [13], [14]. Varios grupos que ya estaban trabajando en el mapeo y localización, en particular en el Massachusetts Institute of Technology [15], Zaragoza [16], [17] el ACFR en Sydney [18], [19], y otros [20], [21], comenzaron a trabajar en serio en SLAM – También llamado "*concurrent mapping and localization*"(CML) en aquel momento – en ambientes interiores, exteriores y submarinos.

En la actualidad el trabajo está enfocado en mejorar el coste computacional y en atajar los errores en la asociación de datos y al cerrar el lazo. El Simposio Internacional de Investigación Robótica de 1999 (ISRR 99) [22] fue un importante punto de encuentro donde se celebró la primera sesión sobre el SLAM y donde se pusieron de acuerdo muchos puntos entre los métodos basados en el filtro de Kalman y los métodos probabilísticos de localización. El taller sobre SLAM de la Conferencia Internacional de Robótica y Automatización del IEEE (ICRA) del año 2000 atrajo a 15 investigadores y se centró en cuestiones tales como la complejidad algorítmica, la asociación de datos y retos para su implementación. El siguiente taller sobre SLAM en el ICRA 2002 atrajo a 150 investigadores con una amplia gama de intereses y aplicaciones. La escuela de verano sobre SLAM de 2002 organizada por Henrik Christiansen en KTH, Estocolmo, atrajo a todos los investigadores principales, junto con unos 50 estudiantes Ph.D. de todo el mundo y fue un tremendo éxito para definir el campo de trabajo. El interés en SLAM ha crecido exponencialmente en los últimos años, y los talleres continúan siendo realizados en las conferencias internacionales ICRA e IROS, organizadas por la sociedad IEEE.

2.1.2. Algoritmos basados en SLAM

La solución al SLAM probabilístico comprende el encontrar una representación apropiada para los modelos de observación y movimiento que permita un cálculo eficiente y consistente de

la estimación y corrección de las distribuciones de probabilidad. La representación más común es, de lejos, en forma de un modelo de espacio de estado con ruido aditivo Gaussiano, que conduce a la utilización del filtro de Kalman extendido (EKF) para resolver el problema SLAM. Una representación alternativa importante es describir el modelo de movimiento del vehículo como un conjunto de muestras de una distribución de probabilidad más general no Gaussiana. Esto conduce a la utilización del filtro de partículas Rao-Blackwellized, o algoritmo FastSLAM, para resolver el problema SLAM. Estos métodos EKF-SLAM y FastSLAM son llamados online SLAM ya que estiman la probabilidad a posteriori en cada posición instantánea del robot, siendo algoritmos de filtrado.

Otros métodos de SLAM son los conocidos como full SLAM. En este caso se busca estimar la posición a posteriori teniendo en cuenta todo el recorrido y no solo la última posición del robot. Estos algoritmos, basados en smoothing, son cada vez más utilizados, debido sobretodo a la inconsistencia de los algoritmos basados en filtros de Kalman por las no linealidades presentes en los entornos reales. Michael Kaes y otros [2] presentan un SLAM basado en *Smoothing and Mapping* con el que obtienen muy buenos resultados de tiempo de cómputo en cada iteración, actualizando el algoritmo propuesto con su versión iSAM2 [23].

2.1.2.1. EKF-SLAM

El algoritmo EKF-SLAM se apoya en la definición de un vector de estados en el que se incluye cada una de las marcas pertenecientes al mapa así como la posición del robot dentro del mapa. Este vector de estados se complementa con una matriz de covarianza en la que se almacena la información sobre la correlación entre la posición del robot y las marcas así como la correlación entre las propias marcas. Los pasos del algoritmo son los clásicos de un filtro de Kalman extendido, añadiendo una etapa de inclusión de marcas nuevas al mapa, en donde se debe calcular la correlación entre la nueva marca vista y el resto de marcas del mapa para seguir teniendo un alto grado de consistencia en el mapa, y añadiendo también la etapa de asociación de datos, en donde se deben emparejar las medidas obtenidas por el robot del entorno con las estimadas del mapa almacenado.

La solución EKF-SLAM es muy conocida y hereda los mismos pros y contras de las soluciones de filtros EKF para problemas de navegación o seguimiento. Cuatro de los puntos clave del algoritmo se discuten en las siguientes líneas.

Convergencia y Consistencia La consistencia es una propiedad de la estimación del estado que consigue congruencia entre dicha estimación del estado y el estado real del entorno. Esto es, el estado real del entorno debe estar dentro del margen establecido por la media y la varianza de la estimación del mismo. Así, una covarianza injustificadamente pequeña dejará fuera de su intervalo de confianza el estado real del entorno. Entonces será imposible que el algoritmo converja a la solución correcta. Esto puede suceder por errores en el cálculo de la matriz de covarianza, ya sean numéricos o lógicos, o por las linealizaciones realizadas, por ejemplo.

Las propiedades de convergencia del EKF-SLAM se encuentran condensadas en varios teoremas enunciados y demostrados en [13] y completados en [24] y [25].

- El determinante de la matriz de covarianza de los objetos del mapa, excluido el robot (y cualquier submatriz principal, por ejemplo la varianza de cada objeto del mapa), es una función monótona no creciente del tiempo (decrece a medida que se incorporan observaciones al filtro).

- Cuando el número de observaciones tiende a infinito, la estimación de los objetos del mapa excepto el robot, tiende a estar totalmente correlada. Esto quiere decir que la distancia relativa entre dos objetos cualesquiera del mapa excepto el robot tiende a ser conocida perfectamente con incertidumbre nula.

- Cuando el número de observaciones tiende a infinito, la covarianza de cada objeto del mapa excepto el robot tiende a un límite inferior definido por la covarianza inicial del robot.

La propiedad de consistencia está íntimamente relacionada con la convergencia. En la tesis [13] se demuestra que para obtener una estimación consistente mediante el filtro de Kalman extendido es necesario mantener la matriz de covarianza completa. El ignorar estas correlaciones lleva automáticamente a la inconsistencia y divergencia del algoritmo.

Los teoremas de convergencia unidos a la propiedad de consistencia antes descrita, concluyen que el algoritmo EKF-SLAM tiende a obtener el estado real del entorno, a medida que se realizan más y más observaciones. No obstante hay que reiterar que este resultado se demuestra suponiendo las linealizaciones despreciables.

Coste Computacional La etapa de observación requiere que se actualicen todas las marcas y la matriz de covarianza cada vez que se realiza una observación del entorno. Esto significa que el coste computacional crece de manera cuadrática con el número de marcas en el mapa. Se ha realizado un intenso esfuerzo para desarrollar variantes eficientes del EKF-SLAM y se han mostrado implementaciones de tiempo real con varios miles de marcas.

Asociación de Datos La formulación estándar del EKF-SLAM es especialmente frágil ante asociaciones incorrectas de marcas. El problema de "cerrar el lazo", en donde el robot vuelve a observar una marca tras un largo recorrido, es especialmente difícil. El problema de asociación se agrava en entornos en donde las marcas no son simples puntos y de hecho cambian en función de la perspectiva desde la que se los ve.

No Linealidad El EKF-SLAM linealiza modelos no lineales de movimiento y observación, con lo que hereda varias limitaciones. La no linealidad es un problema significativo en el EKF-SLAM y lleva inevitable y a veces dramáticamente a la inconsistencia en la solución. Convergencia y consistencia solo se pueden garantizar en el caso lineal.

2.1.2.2. FastSLAM

El algoritmo FastSLAM, introducido por Montemerlo y otros [26], marcó un desplazamiento conceptual fundamental en el diseño del SLAM probabilístico recursivo. Anteriormente los esfuerzos habían ido dirigidos a mejorar el rendimiento del EKF-SLAM a la par que se mantenían sus suposiciones de linealidad y distribución Gaussiana. FastSLAM, basado en el sampling recursivo Monte Carlo, o en filtros de partículas, fue el primero en representar el modelo como no lineal y la distribución de la posición como no Gaussiana.

Esta solución, analizando la estructura del problema SLAM, hace la observación de que si el camino del robot fuera conocido perfectamente, la estimación del mapa sería un problema sencillo y computacionalmente ventajoso, ya que la estimación de la posición de los objetos del entorno en el mapa sería independiente. El estado del mapa sería globalmente observable y la correlación entre los objetos del mapa sería siempre nula.

Otra de las ventajas es que la asociación de datos se realiza para cada partícula, lo que permite la gestión de observaciones ambiguas de una forma mucho más robusta que en el EKF-SLAM. Las asociaciones de datos incorrectas para una partícula evolucionarán en el tiempo provocando la desaparición de la misma, en lugar de la divergencia del filtro.

El coste computacional de esta solución es proporcional al número de partículas empleado y al número de objetos en el mapa. Sin embargo el uso de técnicas de representación eficientes para las estimaciones de los objetos puede reducir el coste computacional hasta crecer solo logarítmicamente con el número de objetos del mapa.

2.1.2.3. iSAM

El algoritmo iSAM [2] usa smoothing para obtener la trayectoria completa y el mapa sin la necesidad de utilizar aproximaciones, explotando la dispersión natural de la matriz de información. La parte principal de esta propuesta es una factorización QR de esta matriz de información. En vez de realizar una refactorización en cada paso, sólo se realiza la factorización QR cada vez que llega una nueva medida.

2.1.3. Tipos de mapa

La construcción de mapas del entorno se ha convertido en uno de los temas que más se han estudiado dentro de la robótica móvil, experimentando un avance notable durante los últimos años. Dotar a un robot móvil de la capacidad de generar un mapa del entorno que le rodea es un paso obligatorio para poder disponer de una total autonomía. Debido a ello se han realizado numerosos trabajos con el objetivo de construir dichos mapas.

En la literatura se pueden encontrar diferentes clasificaciones en los tipos de mapa que se pueden generar. Se pueden clasificar en función del ámbito que abarcan. De este modo se podría diferenciar entre mapas locales, que muestran el entorno cercano al robot o mapas globales, que muestran el entorno completo. Aunque lo más común es diferenciar los mapas en función de su nivel de representación: métrico, topológico y semántico.

2.1.3.1. Nivel de representación métrico

En el nivel de representación métrico el mapa representa la ocupación del entorno a través de medidas numéricas de los objetos y obstáculos presentes en él. Este nivel de representación puede diferenciarse entre geométrico si se representan los obstáculos mediante figuras geométricas, o de rejilla si se descompone el mapa en celdas en las que se considera la probabilidad que éstas estén ocupadas.

Mapas geométricos Los mapas métricos geométricos son sobretodo útiles en interiores, donde la extracción de las características geométricas del entorno es más sencilla. Presentan la ventaja de proporcionar un modelo básico del entorno, facilitando la detección de obstáculos con un coste computacional bajo. El hecho de extraer previamente las características del entorno como los segmentos que generan las paredes o los puntos que generan las esquinas provoca que se filtren los elementos dinámicos de éste.

Los mapas métricos geométricos tienen la desventaja de no proveer de un mapa compacto del entorno, haciendo que su utilización en tareas de navegación de robots autónomos sea escasa ya que un planificador no consideraría todos aquellos obstáculos que no se incluyeron en el mapa al

no haber podido ser asociados a alguna característica geométrica. Del mismo modo su utilización en tareas de exploración es complicada al no conocer qué partes del entorno se han explorado y cuales no. En la figura 2.1 se muestran ejemplos de mapas geométricos de un entorno en un plano de altura constante.

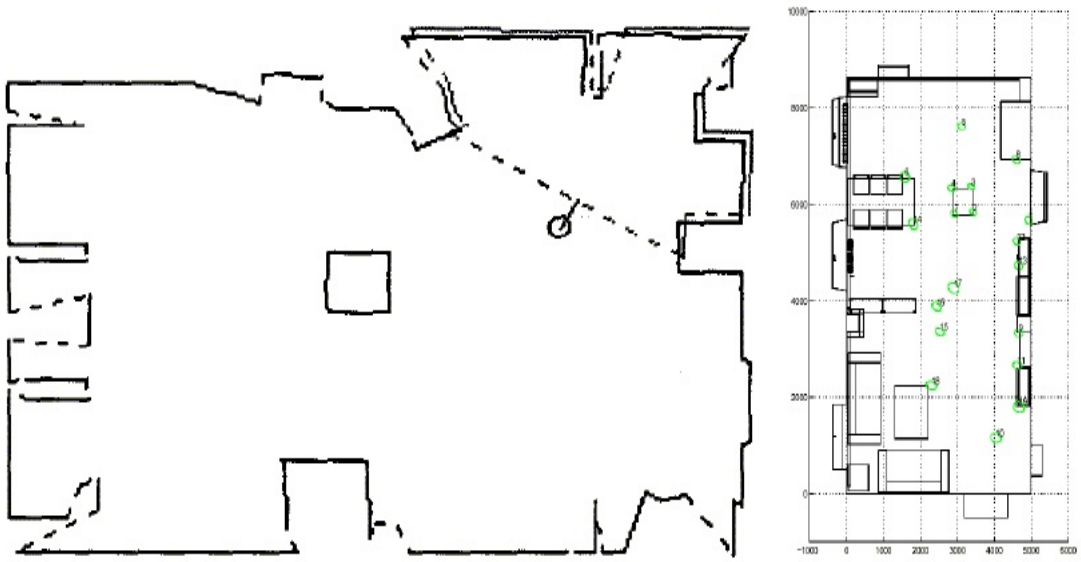

Figura 2.1: Mapas métrico geométrico

Mapas de rejilla Los mapas métricos de rejilla descomponen el entorno en celdas, manteniendo cada una de éstas información de su ocupación. Se conocen también como mapas de ocupación del entorno. Esta información de ocupación se puede representar de manera binaria, esto es ocupada o no, o de un modo probabilístico, donde el grado de ocupación indica la probabilidad de que dicha posición o celda en concreto del entorno se encuentre ocupada. Al contrario que con los mapas métricos geométricos, los mapas métricos de rejilla si proveen de un mapa compacto del entorno. Esto es porque, aunque el espacio se encuentre divididos en celdas, cada una de estas no representa la pertenencia a un objeto en particular. En la figura 2.2 se muestra un ejemplo de mapa métrico de rejilla de un entorno de grandes dimensiones.

Este tipo de mapa tiene la desventaja de requerir mayor capacidad de computación y de almacenamiento cuanto mayor sea el entorno debidas a la propia segmentación de éste. Sin embargo el hecho de que se pueda representar el entorno de manera continua provoca que su uso sea muy útil en tareas de planificación y exploración. Los mapas de ocupación se hicieron populares con los sistemas de localización basados en sensores ultrasónicos, donde las medidas de los mismos son muy ruidosas, requiriendo múltiples medidas para acumular un mapa de ocupación del entorno.

2.1.3.2. Nivel de representación topológico

En el nivel de representación topológico los mapas se basan en las relaciones existentes entre las características del entorno más que en su posición absoluta. En este tipo de mapa se modelan los espacios representativos del entorno y las conexiones entre los mismos, representando los espacios como nodos y sus conexiones entre ellos como arcos. Esta manera de representación

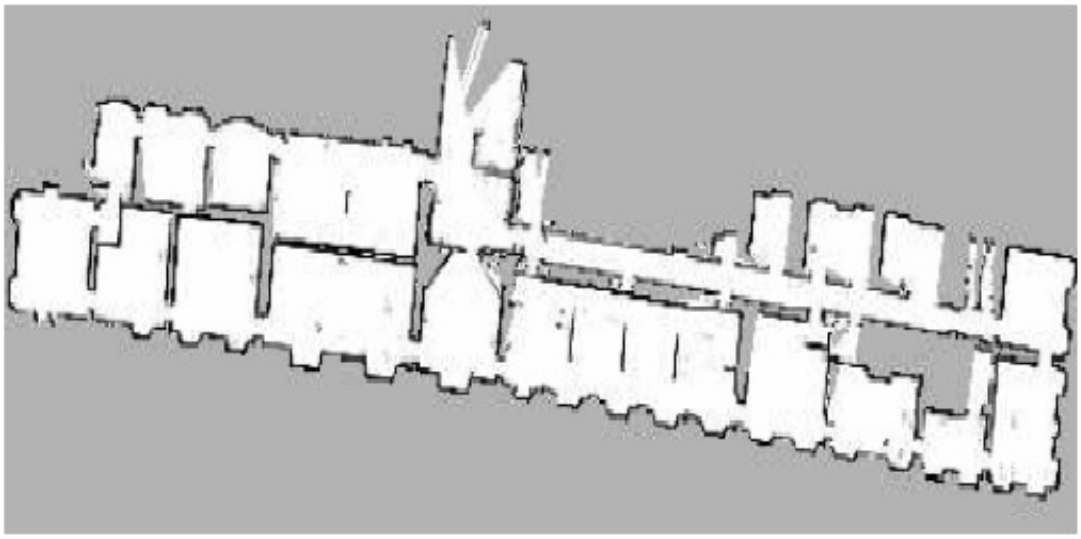

Figura 2.2: Mapas métrico de rejilla

permite construir un grafo del entorno, al que se le puede añadir costes para ir de un nodo a otro. Este tipo de mapa necesita que el robot disponga de un sistema sensorial que permita distinguir los complejos espacios representativos mencionados anteriormente. En la figura 2.3 se muestra un mapa topológico relacionado con un entorno de interior. Los nodos representan estancias del entorno y las conexiones entre los nodos del grafo representan las posibles comunicaciones o caminos entre las diferentes estancias.

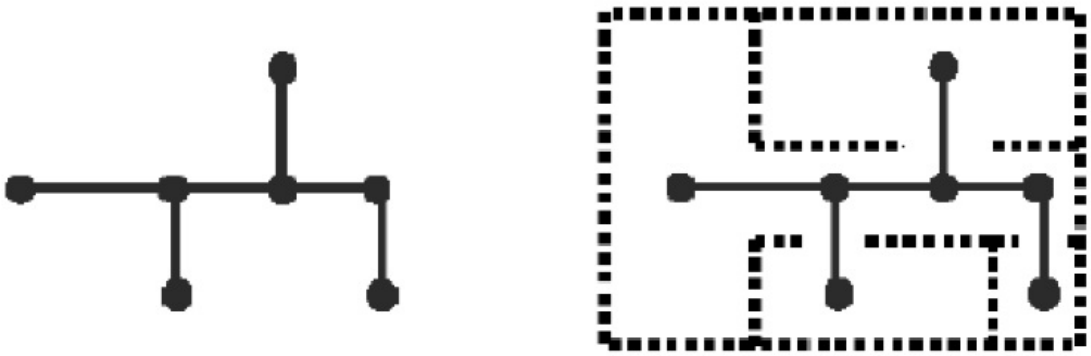

Figura 2.3: Mapa topológico

2.1.3.3. Nivel de representación semántico

El nivel de representación semántico se diferencia del topológico en que toda información geométrica queda descartada. De este modo los lugares y las conexiones más representativas del entorno se clasifican mediante etiquetas con atributos lingüísticos. De esta forma se obtiene un resultado mucho más similar a la comunicación humana que en los anteriores modelos. Una forma de tratar de dar semántica a un mapa es intentando identificar clases de lugares, por ejemplo con algoritmos de agrupamiento. Un ejemplo de descripción semántica de un entorno con obstáculos se muestra en la figura 2.4

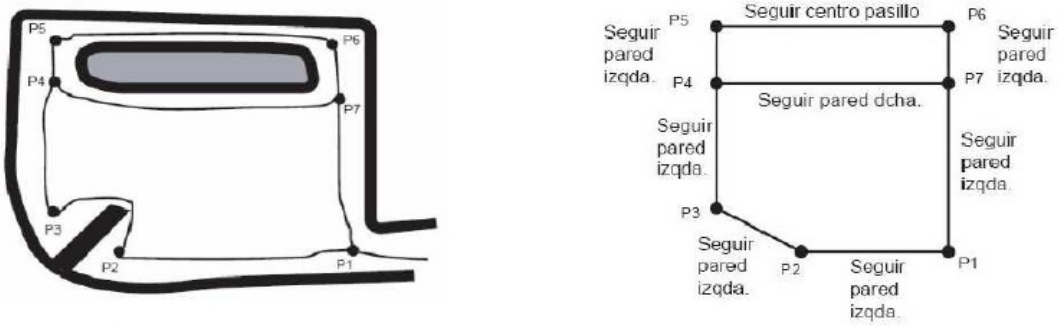

Figura 2.4: Mapa semántico

2.1.4. ROS - Robot Operating System

Figura 2.5: ROS

El presente trabajo fin de máster se ha desarrollado dentro del entorno proporcionado por la plataforma ROS-Robot Operating System. ROS es un marco de trabajo para desarrollo de aplicaciones de robótica, que intenta proveer de funcionalidades similares a las de un sistema operativo en un conjunto heterogéneo de ordenadores. El desarrollo de ROS comenzó en el año 2007 bajo el nombre de Switchyard por el laboratorio de inteligencia artificial de la Universidad de Stanford, como apoyo al proyecto STAIR. En el año 2008 el desarrollo de ROS continuó en Willow Garage, un instituto de investigación de robótica, con más de veinte instituciones colaborando en el proyecto.

En la actualidad el proyecto ROS se concentra en `www.ros.org`, donde una amplia comunidad de investigadores y programadores desarrollan sistemas para ser integrados en ROS.

ROS provee servicios estándar de sistema operativo tales como abstracción del hardware, control de dispositivos a bajo nivel, implementación de funcionalidades comunes, paso de mensajes entre procesos y gestión de paquetes. Está basado en una arquitectura gráfica en la que el procesado tiene lugar en los nodos, que podrán recibir, enviar y multiplexar mensajes de sensores, control, estado, planificación, actuación y cualquier otro tipo definido por el usuario. La librería se ha desarrollado sobre sistemas basados en Unix, como Ubuntu.

ROS tiene dos caras. La cara de sistema operativo descrita anteriormente y ros-pkg, un conjunto de paquetes desarrollados por la comunidad de usuarios que hay detrás de ROS que implementan funcionalidades como SLAM, planificación de recorridos, percepción, simulación, etc.

ROS se ha publicado bajo los términos de la licencia BSD y es un software de código abierto. Es libre para uso tanto comercial como de investigación. Los paquetes de ros-pkg están bajo una variedad de licencias de código abierto.

ROS dispone de varios paquetes con propuestas al problema de la generación de mapas y localización. Estos paquetes hacen uso de dispositivos provistos de láser de barrido o del Kinect de Microsoft como sistemas sensoriales. Los paquetes que se pueden encontrar usando cámaras como sistema sensorial, ya sea con una o varias cámaras, aun están en fase de desarrollo.

2.2. Fundamentos teóricos del iSAM

Smoothing en el contexto del SLAM se refiere comunmente al problema del full SLAM [27]. Está íntimamente relacionado con el *bundle adjustment* [28] y con *Structure From Motion*. La primera implementación [29] se basó en la inversión de matrices. A partir de ahí varias aproximaciones a la solución del SLAM con *Smoothing and Mapping* se han implementado, como LRGC [30], [31], Atlas [32], Graphical SLAM [33], multi-level relaxation [34], square root SAM [35], GraphSLAM [27], y Treemap [36], [37].

2.2.1. Smothing and Mapping (SAM)

En esta sección se revisa la formulación del problema del SLAM en el contexto de *Smoothing*, pero enfocándose en una solución basada en factorización de matrices QR. Se comienza con el modelo probabilístico subyacente al enfoque de *Smoothing* para la solución del SLAM, y se muestra cómo la inferencia en este modelo deriva en un problema de mínimos cuadrados. Entonces se puede obtener una formulación lineal equivalente en modelo matricial linearizando las funciones de medida. Por último se provee de una solución basada en factorización de matrices QR.

2.2.1.1. Modelo probabilístico para el SLAM

Se formula el problema del SLAM en términos de un modelo de red bayesiana mostrado en la figura 2.6. Se representa el estado del robot en el tiempo i^{th} por x_i, con $i \epsilon 0 \dots M$, una marca del mapa como l_j, con $j \epsilon 0 \dots N$, las entradas de control como u_i, con $i \epsilon 0 \dots M$ y una medida como z_k con $k \epsilon 0 \dots K$. La fórmula de probabilidad conjunta se representa por:

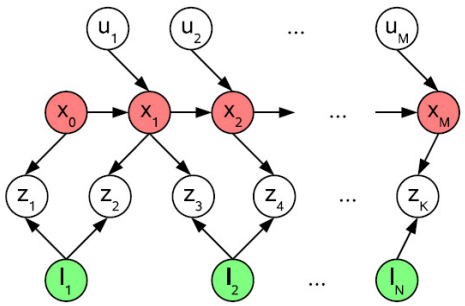

Figura 2.6: Representación del problema del SLAM como una red bayesiana

$$P(X, L, U, Z) = P(x_0) \prod_{i=1}^{M} P(x_i|x_{i-1}, u_i) \prod_{k=1}^{K} P(z_k|x_{i_k}, l_{j_k}) \tag{2.1}$$

donde $P(x_0)$ es la probabilidad a priori del estado inicial, $P(x_i|x_{i-1}, u_i)$ es el modelo de movimiento parametrizado por la entrada de control u_i, y $P(z_k|x_{i_k}, l_{j_k})$ es el modelo de medición de marcas, asumiendo que se conocen las correspondencias (i_k, j_k) para cada medida z_k. Se asume que los modelos de movimiento y medida son modelos gaussianos.

El modelo de movimiento

$$x_i = f_i(x_{i-1}, u_i) + w_i \tag{2.2}$$

describe el movimiento del robot como respuesta a la entrada de control, donde w_i es una medida de ruído gaussiano de media 0 y matriz de covarianza Λ_i.

El modelo de medida

$$z_k = h_k(x_{i_k}, k_{j_k}) + v_k \tag{2.3}$$

modela el sistema sesorial del robot, donde v_k es una medida de ruído gaussiano de media 0 y matriz de covarianza Σ_k.

2.2.1.2. SLAM como un problema de mínimos cuadrados

Obtener una estimación óptima para el grupo de incógnitas dadas todas las medidas disponibles supone convertir el problema en una formulación equivalente de mínimos cuadrados basada en una estimación *Maximum a Posteriori* (MAP). Al realizar Smoothing en vez de filtrado, se busca la estimación *Maximum a Posteriori* de la trayectoria completa $X = \{x_i\}$ y el mapa de marcas $L = \{l_j\}$, dadas las medidas $Z = \{z_k\}$ y las entradas de control $U = \{u_i\}$.

Agrupando todas las variables aleatorias de X y L en el vector $\theta = (X, L)$, se obtiene la estimación MAP θ^* minimizando el logaritmo negativo de la función de probabilidad conjunta $P(X, L, Z)$ a partir de 2.1:

$$\theta^* = \arg\min_{\theta} - \log P(X, L, Z) \tag{2.4}$$

Combinado con el modelo de movimiento y medida, esto lleva al siguiente problema de mínimos cuadrados no lineales:

$$\theta^* = \arg\min_{\theta} \left\{ \sum_{i=1}^{M} \|f_i(x_{i-1}, u_i) - x_i\|^2_{\Lambda_i} + \sum_{k=1}^{K} \|h_k(x_{i_k}, l_{j_k}) - z_k\|^2_{\Sigma_k} \right\} \tag{2.5}$$

donde se usa la notación $\|e\|^2_{\Sigma} = e^T \Sigma^{-1} e$ para el cuadrado de la distancia de Mahalanobis dada una matriz de covarianza Σ.

En la práctica se considera una versión linearizada del problema. Si el modelo de movimiento f_i y el modelo de medida h_k son no lineales y no existe un buen punto de linearización, los métodos de optimización no lineal resuelven una sucesión de aproximaciones a esta ecuación con el objetivo de acercarse al mínimo. Por lo tanto se lineariza el problema de mínimos cuadrados asumiendo que o bien hay un buen punto desde el que realizar la linearización o que se está realizando una iteración de un método de optimización no lineal:

$$\delta\theta^* = \arg\min_{\delta\theta} \left\{ \sum_{i=1}^{M} \left\| F_i^{i-1} \delta x_{i-1} + G_i^i \delta x_i - a_i \right\|^2_{\Lambda_i} + \sum_{k=1}^{K} \left\| H_{i_k} \delta x_{i_k} + J_k^{j_k} \delta l_{j_k} - c_k \right\|^2_{\Sigma_k} \right\} \tag{2.6}$$

donde $H_k^{i_k}$ y $J_k^{j_k}$ son los Jacobianos de h_k respecto a un cambio en x_{i_k} y l_{i_k} respectivamente, F_i^{i-1} es el Jacobiano de f_i en x_{i-1} y $G_i^i = I$ por simetría. a_i y c_k son los errores de las predicciones de las medidas de odometría y observación de las marcas respectivamente.

Agrupando todas las matrices de Jacobianos en una sola matriz A, y todos los vectores en un vector tipo *right-hand side* (rhs) b, se obtiene un problema estándar de mínimos cuadrados:

$$\theta^* = \arg\min_{\theta} \|A\theta - b\|^2 \tag{2.7}$$

donde el vector $\theta \epsilon \mathbb{R}^n$ contiene todas las variables de posición y marcas, la matriz $A \epsilon \mathbb{R}^{mxn}$ es el Jacobiano de la medida, matriz la cual es dispersa, con m filas de medidas, y $b \epsilon \mathbb{R}^m$ es el vector rhs. Este tipo de sistemas de mínimos cuadrados se convierten en sistemas de ecuaciones lineales igualando la derivada de $\|A\theta - b\|^2$ a 0, resultando en las habituales ecuaciones $A^T A\theta = A^T b$. Este sistema de ecuaciones se puede resolver por descomposición Cholesky de $A^T A$.

2.2.1.3. Resolución por factorización QR

Se aplica la factorización de matrices QR estándar [38] para resolver el problema de mínimos cuadrados 2.7. En el caso lineal, el Jacobiano de la medida A de este problema de mínimos cuadrados es independiente de la estimación actual θ, por lo que se puede reescribir el problema de mínimos cuadrados como:

$$\left\| Q \begin{bmatrix} R \\ 0 \end{bmatrix} \theta - b \right\|^2 = \|R\theta - d\|^2 + \|e\|^2 \tag{2.8}$$

donde Q es una matriz ortogonal y se define $[d, e]^T \triangleq Q^T b$. Observar que si el Jacobiano A es una matriz mxn, entonces Q y R son matrices mxm y nxn respectivamente. El primer término $\|R\theta - d\|^2$ desaparece de la solución de mínimos cuadrados θ^* dejado el segundo término $\|e\|^2$ como el residual del problema de mínimos cuadrados.

La solución para la trayectoria completa del robot así como el mapa de marcas se puede recuperar de manera eficiente en cualquier momento durante el proceso de mapeo. Esto se consigue realizando back-substitution usando el factor actual R y el vector rhs d para obtener una actualización de todas las variables θ del modelo basadas en:

$$R\theta = d \tag{2.9}$$

Mientras que la complejidad del back-substitution es de $O(n^2)$ para matrices densas, es más eficiente en este caso. Se asume tal como proponen en [2] que las entradas por columna en R no dependen del número de variables n que definen el mapa y la trayectoria. Incluso si hay una dependencia con n en caso de entornos muy repetitivos, esta dependencia típicamente es tan pequeña que se puede considerar inexistente. Bajo este supuesto, la complejidad de back-substitution sería $O(n)$.

2.2.1.4. Incremental SAM (iSAM)

La solución incremental al problema del full SLAM [27] se basa en la actualización de una representación factorizada de la matriz de información del problema de mínimos cuadrados de 2.7. Por simplicidad se considera primero el caso de modelo de movimiento y medida lineal para volver al modelo no lineal más adelante.

Rotaciones de Givens Una aproximación estándar para obtener la factorización QR del Jacobiano de la medida A utiliza las *rotaciones de Givens* [38] para eliminar todas las entradas por debajo de la diagonal principal, una a una. Como se verá más adelante esta aproximación

fácilmente se extiende a las actualizaciones de factorización, como será necesario al incorporar nuevas medidas. El proceso comienza con la entrada no nula más abajo y a la izquierda posible y procede por columnas o por filas aplicando la rotación de Givens

$$\begin{bmatrix} \cos\phi & \sin\phi \\ -\sin\phi & \cos\phi \end{bmatrix} \tag{2.10}$$

a las filas i y k con $i > k$. El parámetro ϕ se elige de modo que la entrada (i, k) de A se vuelva 0 tal como se muestra en la figura 2.7. Una vez todas las entradas por debajo de la diagonal principal se han hecho 0, todas las entradas contienen el factor R. Hay que tener en cuenta que un Jacobiano A de la medida que sea disperso dará como resultado un factor R disperso, al menos para un ordenado adecuado de variables. Sin embargo la matriz ortogonal de rotación Q es típicamente densa, y este es el motivo por el que nunca se almacena de manera explicita o incluso se construye en la práctica. En cambio, es suficiente con actualizar la matriz rhs b con las mismas rotaciones que se aplicaron a A. En vez de factorizar el Jacobiano de la medida actualizado cada

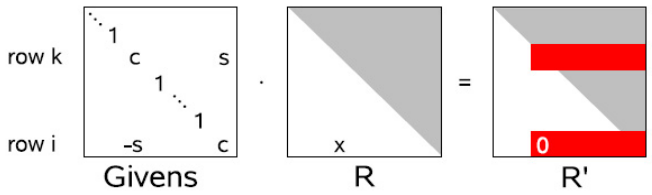

Figura 2.7: Uso de la rotación de Givens como paso para transformar una matriz general en otra con valores no nulos solo en la parte superior de la diagonal principal. El valor marcado como 'x' se elimina

vez que llega una nueva medida, es más eficiente modificar la factorización previa a partir de actualización QR. Añadir una nueva fila de medida w^T y matriz rhs γ al factor actual R y matriz rhs d desemboca en un nuevo sistema que no está en su forma correctamente factorizada:

$$\begin{bmatrix} Q^T & \\ & 1 \end{bmatrix} \begin{bmatrix} A \\ w^T \end{bmatrix} = \begin{bmatrix} R \\ w^T \end{bmatrix} \qquad rhs : \begin{bmatrix} d \\ \gamma \end{bmatrix} \tag{2.11}$$

Nótese que este sería el mismo sistema que se obtendría aplicando las rotaciones de Givens para eliminar todas las entradas por debajo de la diagonal principal, excepto las de la última fila, que se corresponde a la nueva medida. Por lo tanto se pueden determinar las rotaciones de Givens para transformar en cero esta nueva fila, generando el nuevo factor actualizado R'. Del mismo modo que para la factorización completa, se actualiza de manera simultánea el rhs con las mismas rotaciones para obtener d'. Se muestran varios pasos de este proceso de actualización en la figura 2.8.

Para iSAM, la actualización QR es eficiente. En general, el máximo número de rotaciones de Givens necesarias para añadir una nueva fila es n. Sin embargo, como R y la nueva fila que se corresponde con la medida son dispersas, sólo se necesita un número constante de rotaciones de Givens. Además, nuevas medidas se refieren de manera típica a variables añadidas recientemente, con lo que lo más común es encontrarse con que sólo la parte más a la derecha de la fila que se corresponde a la última medida está poblada (de manera dispersa). Un ejemplo demostrando la localidad del proceso de actualización se muestra en la figura 2.8.

Es sencillo añadir nuevas marcas y posiciones a la factorización QR, ya que se puede expandir el factor R por el número apropiado de columnas y filas con valor 0, antes de actualizarlos con

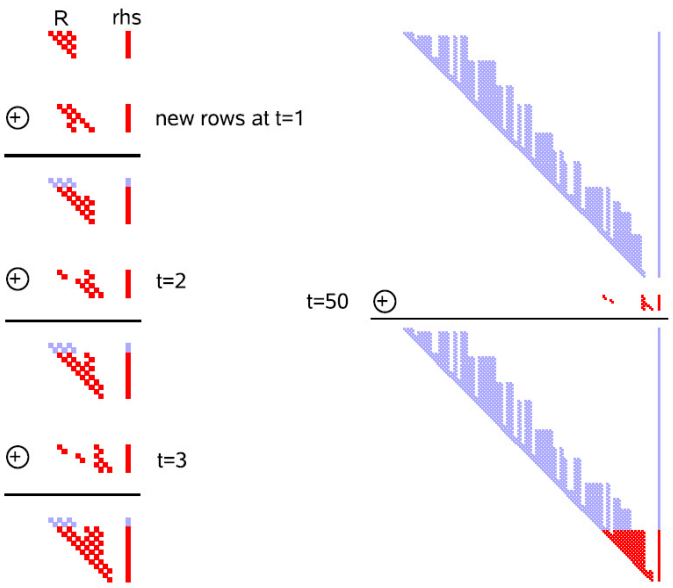

Figura 2.8: Actualización de la representación factorizada de la matriz de información para un ejemplo de tarea de exploración

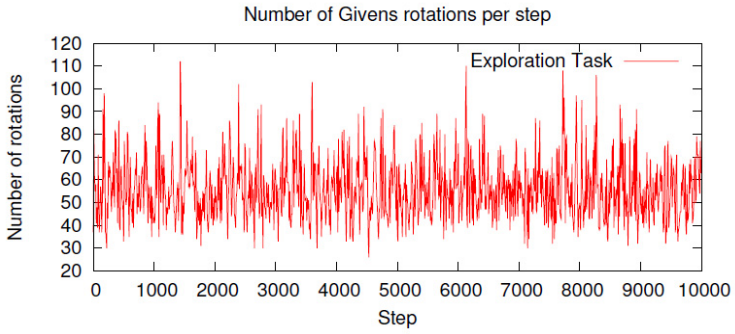

Figura 2.9: Cantidad de rotaciones de Givens necesaria por iteración para una simulación de tarea de exploración

las nuevas filas correspondientes a las medidas. De manera similar el *rhs* d se aumenta con el mismo número de entradas nulas.

El ordenado de las matrices requiere de cierta consideración. En cada paso se añaden primero las marcas para la posición actual y solo después se añade la nueva posición. Esto asegura que la última variable en el sistema es la última posición, que es necesario para recuperar de un modo eficiente las covarianzas.

Para una tarea de exploración en el caso lineal, el número de rotaciones que se necesitan incorporar al set de nuevas marcas y medidas de odometría es independiente del tamaño de la trayectoria y del mapa [39]. Con lo que la tarea de actualización tiene complejidad $O(1)$ para tareas de exploración. Recuperar todas las variables tras cada paso requiere un tiempo $O(n)$, pero sigue siendo muy eficiente tras 10000 pasos, a aproximadamente 0.12 segundos por paso. Además, se puede obtener una aproximación de tiempo constante, ya que solo las variables más recientes cambian lo suficiente como para justificar un recálculo. Esto se puede conseguir parando el back-substitution una vez el cambio en la variable baja de un cierto umbral. En la figura 2.9 se muestran la cantidad de rotaciones necesitadas por cada iteración que, tal como se

ha adelantado, resulta ser independiente al tamaño del mapa.

Lazos y reordenado de variables Los entornos con lazos no cumplen con la propiedad de actualizaciones locales, resultando en un aumento de la complejidad. En contraste con la exploración pura, donde las marcas solo se ven en pasos consecutivos, los lazos en la trayectoria del robot lo llevan a una posición visitada con anterioridad. Este hecho introduce correlaciones entre la posición actual y marcas observadas con anterioridad, que a su vez están conectadas con posiciones anteriores en la trayectoria del robot. En la figura 2.10 se muestran resultados basados en un entorno simulado con varios lazos.

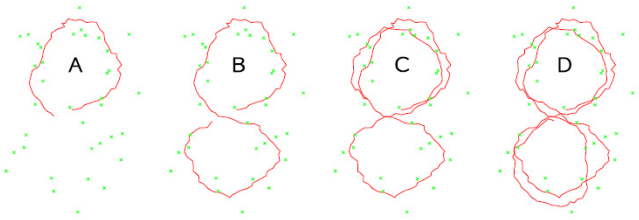

Figura 2.10: Entorno simulado.

Los lazos pueden llevar a un aumento significativo de la complejidad computacional debido a un aumento de factores no nulos en la matriz R. Estas entradas no nulas más allá del patrón de dispersión de la matriz de información se llaman $fill-in$. Aunque la matriz de información se mantiene dispersa en el proceso, la actualización incremental del factor R lleva al $fill-in$ como se puede observar en la figura 2.11. Este $fill-in$ es local y no afecta a la exploración futura, como es evidente para el problema. Sin embargo este $fill-in$ se puede evitar, ya que depende del ordenado de las variables. A pesar de que obtener el mejor ordenado es computacionalmente duro, se han desarrollado heurísticos eficientes como COLAMD (column approximate minimum degree) [40] en álgebra lineal que obtienen buenos resultados para el problema del SLAM tal como se evalúa en [35]. El mismo factor R tras el reordenado muestra signos de $fill-in$ que se pueden considerar despreciables, como se puede ver en 2.11. Aún así, el reordenado de variables y

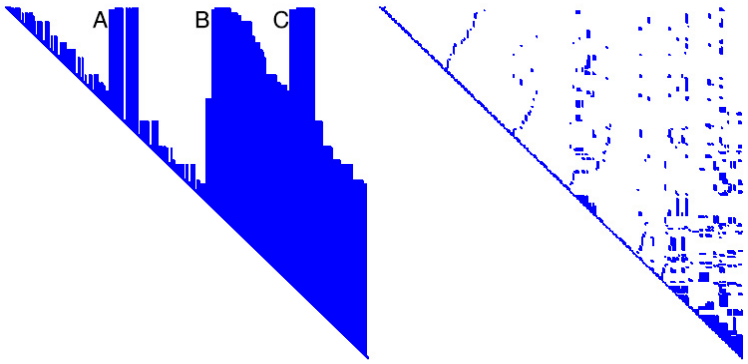

Figura 2.11: Factor R con fill-in previo a la reodrenación vs posterior a la reordenación de variables.

la posterior factorización del nuevo Jacobiano de la medida es computacionalmente lenta cuando se realiza en cada paso. Por lo tanto se proponen actualizaciones rápidas seguidas de reordenados periódicos, como se muestra en la figura 2.12. Cuando el robot observa de manera continua las mismas marcas, como por ejemplo ocurre en habitaciones pequeñas, esta aproximación fallará ya

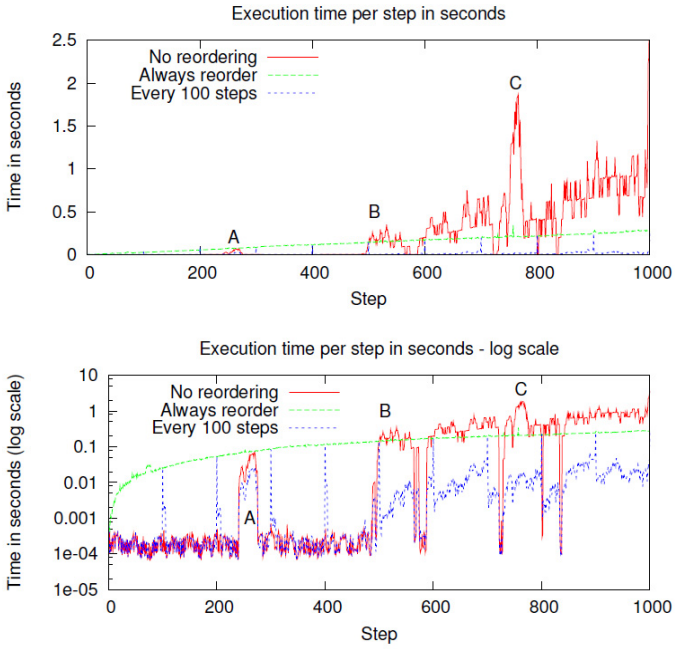

Figura 2.12: Tiempo de ejecución por iteración para diferentes estrategias de actualización.

que la matriz de información se volverá densa. De todas formas en este caso los filtros también fallarán debido a que subestimarán las incertidumbres que finalmente convergerán a 0. Una mejor solución para este tipo de escenario es, llegado el punto de que se conoce el entorno, cambiar a un algoritmo de localización.

Sistemas No Lineales Actualizar el punto de linealización basado en una nueva estimación de la variable cambia el Jacobiano de la medida. Una manera entonces para obtener una nueva factorización es refactorizar el Jacobiano de la medida. Sin embargo, en muchas situaciones no es necesario para llevar a cabo relinearización [41]. En primer lugar, las mediciones son normalmente bastante exactas a escala local, por lo que no es necesario que se relinearice en cada paso. Por lo tanto, para evitar el cálculo adicional, la relinearizacion se puede realizar junto con la reordenación periódica variable. En segundo lugar, las mediciones también son locales y sólo afectan a un pequeño número de variables directamente, con sus efectos disminuyendo rápidamente mientras que se propaga a través del grafo. Una excepción es una situación tal como un cierre de lazo, que puede afectar a muchas variables a la vez. En cualquier caso, es suficiente para llevar a cabo relinearizacion selectiva sólo sobre variables cuya estimación ha cambiado en más de un cierto umbral. La relinearizacion entonces también se puede aplicar de forma incremental.

Capítulo 3

Objetivos

3.1. Objetivos

Se dispone de una solución [1] al problema del SLAM basada en la creación de un mapa probabilístico de marcas naturales del entorno, capturadas a partir de un sistema de visión estereoscópica a bordo de un robot. Se pretende optimizar el funcionamiento de dicha solución, cambiando el enfoque del EKF SLAM del que se parte a uno basado en *Smoothing and Mapping*.

El principal objetivo es la mejora del coste computacional del algoritmo, permitiendo pasar de entornos muy pequeños (menos de 300 marcas en el entorno) a un entorno más grande. También se busca que el algoritmo pueda funcionar para aplicaciones que requieran de manera inmediata la información de la posición actual del robot.

Se presentará por tanto el funcionamiento del algoritmo propuesto para la solución del SLAM a partir de iSAM y se compararán los resultados obtenidos con el enfoque de EKF SLAM.

Los objetivos específicos de este trabajo son por tanto:

1. Realizar un sistema de visión que devuelva como salida la posición de un grupo de marcas naturales del entorno. Las características del sistema serán:
   - Se dispone de un sistema de visión estereoscópica calibrado que entregue bien un par de imágenes estéreo, o bien un par de imagen más mapa de disparidad, así como los parámetros de calibración del sistema de visión estereoscópica.
   - El sistema debe proporcionar marcas estables en el tiempo.
   - El sistema debe proporcionar marcas de una manera repetible.
2. Solucionar el problema del SLAM con un estimador bayesiano, permitiendo obtener la posición y orientación de un robot móvil, al mismo tiempo que se genera un mapa del entorno. Las siguientes restricciones se proponen para el estimador:
   - Las ecuaciones que definen el sistema son no lineales.
   - Se dispone de un modelo de movimiento basado en el uso de sensores de odometría internos del robot.
3. El estimador debe mejorar las prestaciones del estimador basado en el EKF.
4. La solución planteada debe poder entregar una solución a la localización del robot en un tiempo lo suficientemente pequeño como para poder realizar tareas de navegación online.
5. Comparación de resultados entre el iSAM y el SLAM con EKF y posterior demostración de la mejora de resultados en cuanto al tiempo de procesado y el tamaño del mapa al cambiar la solución al SLAM por el enfoque usando *Smoothing and Mapping*.

Capítulo 4

Desarrollo

4.1. Modelo matemático del sistema

En este punto se detalla el modelado matemático del movimiento y sistema de observación incluido en el robot.

Notación empleada

Los valores escalares se representan en tipografía normal, mayúsculas o minúsculas. Los vectores se representan mediante letras minúsculas en negrita ($\mathbf{x}$) y las matrices mediante letras mayúsculas en negrita ($\mathbf{P}$)

4.1.1. Representación matemática del sistema

En este proyecto se parte de la simplificación de que el robot se mueve en un plano conocido. Se define un origen de coordenadas global o del mundo W con respecto al cual se almacenarán los puntos tridimensionales $\mathbf{p}$ que componen el mapa. El origen W es un origen estático, al igual que el mapa, con lo cual, la posición del robot variará con el tiempo si se representa con respecto a W. Puesto que el robot se supone de movimiento en un plano, será suficiente con representar la posición del mismo en dos coordenadas, en este caso los ejes de coordenadas x e y del origen de coordenadas W. Es decir, el movimiento del robot se realiza en el plano $z = 0$, o un plano paralelo a este. La orientación del robot se presenta mediante un simple ángulo θ, que representa la cantidad de rotación con respecto a un eje de rotación paralelo al eje z. En la figura 4.1 se muestra la distribución geométrica del problema, donde se muestra el origen W, el robot asociado a su posición y un punto del entorno.

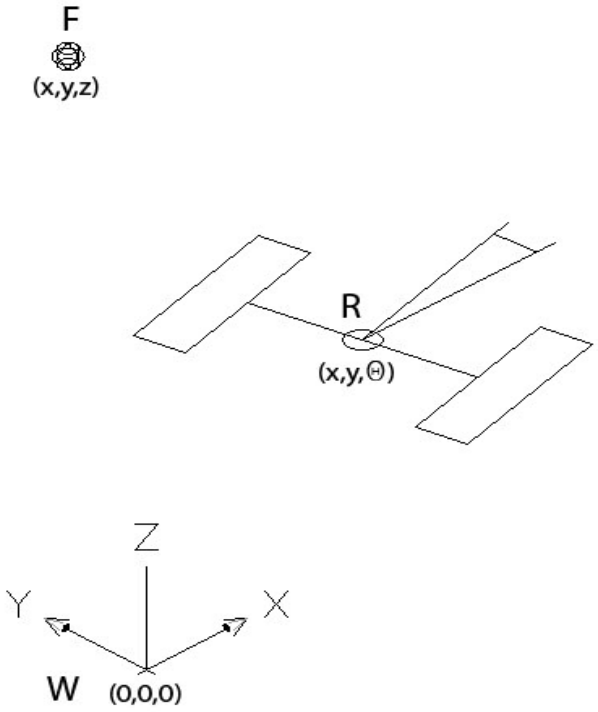

Figura 4.1: Sistema de coordenadas

El vector de estado de un objeto $\mathbf{p}$ del mapa se representa por el vector:

$$\mathbf{x}_F = \begin{bmatrix} x & y & z \end{bmatrix}^T \tag{4.1}$$

El vector de estado de cada una de las posiciones del robot R en el mapa se representa por el vector:

$$\mathbf{x}_R = \begin{bmatrix} x & y & \theta \end{bmatrix}^T \tag{4.2}$$

A diferencia del caso del filtro de Kalman extendido, en el que se representaba el conjunto de marcas y la posición del robot como un vector de estados del sistema y su matriz de varianzas asociada, ahora se representa el mapa como un grafo en donde hay dos tipos de nodos y dos tipos de arcos que los unen. Los nodos de posiciones consecutivas del robot se conectan por un arco de movimiento, mientras que los nodos de medición de marcas se conectan con un nodo de posición del robot mediante un arco de medición. Este grafo se puede ver representado en la figura 2.6.

4.1.2. Estructura del robot

Las características físicas del robot deben tenerse en cuenta a la hora de resolver el problema. En el caso que ocupa este proyecto se dispone de un robot con un modelo de tracción diferencial, del que se dispone información del movimiento de las ruedas mediante el uso de sensores de odometría. También se encuentran instaladas dos cámaras en una estructura sobre la base del robot que harán la función de sensores del entorno. En la figura 4.2 se muestra una imagen de la plataforma robótica utilizada.

Figura 4.2: Robot Qbo

Las características técnicas de la plataforma se muestran a continuación:

- PC incorporado con un procesador Intel i3, 2GB de RAM y un disco duro SATA de 500GB.

- Base del robot con dos ruedas que permiten movimiento con configuración diferencial.
- Cabeza del robot que incluye dos WebCam VGA con conector USB y un sistema de movimiento "pan and tilt".
- Placas de control que dan acceso al sistema incorporado de estimación de la posición a partir de la odometría de las ruedas.
- Posibilidad de incluir las medidas del sistema de control inercial a la odometría del robot mediante un filtro de Kalman.
- Tres micrófonos y dos altavoces para permitir el uso de sistemas de reconocimiento y síntesis de voz.

4.1.2.1. Modelo cinemático del robot

Un robot de tracción diferencial dispone de dos ruedas motrices y uno o más puntos de apoyo para estabilizarlo. La traslación y rotación del robot son función del movimiento de las dos ruedas motrices. Si se considera el punto medio entre las dos ruedas motrices como la posición del robot en un sistema de coordenadas global, las ecuaciones que definen el movimiento del robot sobre ese sistema de coordenadas son:

$$\begin{aligned} \dot{x} &= v(t)\cos(\theta(t)) \\ \dot{y} &= v(t)\sin(\theta(t)) \\ \dot{\theta} &= w(t) \end{aligned} \tag{4.3}$$

Donde $\dot{x}$, $\dot{y}$ y $\dot{\theta}$ son la derivada con respecto al tiempo de x, y y θ. $\dot{x}$, $\dot{y}$ y $\dot{\theta}$ representan la velocidad lineal y angular del robot.

Para determinar la posición del robot en un instante dado habrá que integrar las velocidades con respecto al tiempo:

$$\begin{aligned} x &= x_0 + \textstyle\int_{\Delta t} v(t)\cos(\theta(t))dt \\ y &= y_0 + \textstyle\int_{\Delta t} v(t)\sin(\theta(t))dt \\ \theta &= \theta_0 + \textstyle\int_{\Delta t} w(t)dt \end{aligned} \tag{4.4}$$

Suponiendo que se integra sobre un intervalo de tiempo infinitésimamente pequeño, se pueden simplificar las integrales por desplazamientos diferenciales, quedando las ecuaciones de la posición del robot de la siguiente manera:

$$\begin{aligned} x &= x_0 + \Delta x \\ y &= y_0 + \Delta y \\ \theta &= \theta_0 + \Delta\theta \end{aligned} \tag{4.5}$$

Suponiendo que se realizan muestreos de la posición del robot en intervalos muy cortos de tiempo, se puede discretizar las funciones con respecto a instantes de tiempo k discretos:

$$\begin{aligned} x(k) &= x(k-1) + \Delta x(k) \\ y(k) &= y(k-1) + \Delta y(k) \\ \theta(k) &= \theta(k-1) + \Delta\theta(k) \end{aligned} \tag{4.6}$$

Al suponer un intervalo de tiempo infinitésimamente pequeño entre muestra y muestra, se puede suponer también que la velocidad angular de las ruedas motrices se mantiene constante entre muestra y muestra. Este hecho simplifica mucho las ecuaciones que modelan el desplazamiento del robot durante cada instante de tiempo. En la figura 4.3 $V(k)$ es el desplazamiento del centro de movimiento del robot en un intervalo $[k-1, k]$ y $\Phi(k)$ la rotación que ha experimentado el robot en el mismo intervalo de tiempo. Estos desplazamientos se pueden obtener según:

$$\Phi(k) = \frac{\Delta S_{k,R} - \Delta S_{k,L}}{d} \tag{4.7}$$

$$V(k) = \frac{\Delta S_{k,R} + \Delta S_{k,L}}{2} \tag{4.8}$$

donde $\Delta S_{k,R}$ y $\Delta S_{k,L}$ son la distancia recorrida por la rueda derecha e izquierda respectivamente y d representa la distancia entre las dos ruedas. Esos desplazamientos se pueden obtener a partir de la lectura de los sensores de cada una de las ruedas:

$$\Delta S = \frac{2\pi r}{n} \Delta N \tag{4.9}$$

donde r es el radio de las ruedas, n el número de pulsos por vuelta, ΔN la cantidad de pulsos que se ha movido la rueda en el periodo de tiempo medido y ΔS el desplazamiento de la rueda en dicho periodo.

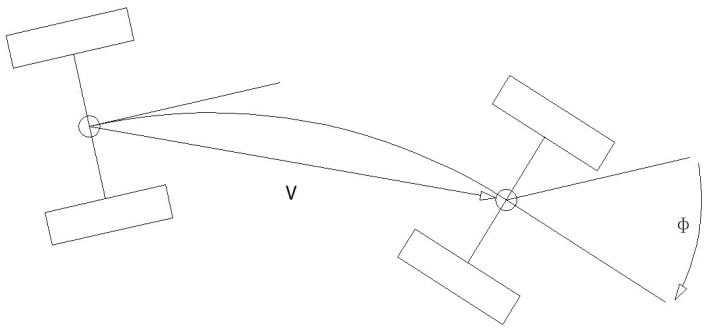

Figura 4.3: Movimiento del robot

La diferencia en la posición angular que experimenta el robot en un intervalo de tiempo es la misma que se obtiene en la ecuación 4.7, mientras que el desplazamiento en el plano definido por el sistema de coordenadas se obtiene a partir de la figura 4.4, resultando

$$\Delta x(k) = V(k) \cos\left(\theta(k-1) + \Phi(k)\right) \tag{4.10}$$

$$\Delta y(k) = V(k) \sin\left(\theta(k-1) + \Phi(k)\right) \tag{4.11}$$

Si despejamos los resultados de las ecuaciones 4.10 y 4.11 en la ecuación 4.6 obtendremos el siguiente sistema de ecuaciones

$$\begin{aligned} x(k) &= x(k-1) + V(k) \cos\left(\theta(k-1) + \Delta\theta(k)\right) \\ y(k) &= y(k-1) + V(k) \sin\left(\theta(k-1) + \Delta\theta(k)\right) \\ \theta(k) &= \theta(k-1) + \Delta\theta(k) \end{aligned} \tag{4.12}$$

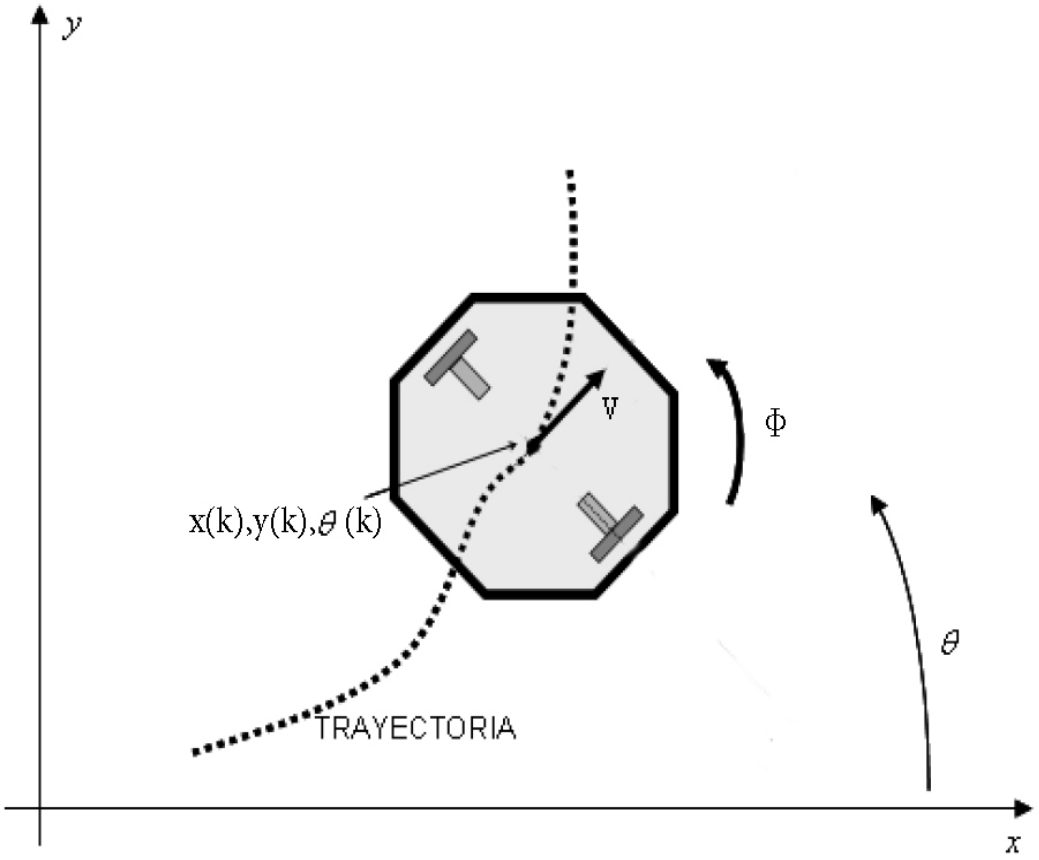

Figura 4.4: Movimiento del robot

Este sistema de ecuaciones representa el movimiento del robot sobre el plano en función de su desplazamiento lineal y angular. Se puede observar en el sistema de ecuaciones que el robot se modela como la siguiente planta no lineal a través de la función vectorial $\mathbf{f}_R : \mathbf{R}^3 \times \mathbf{R}^2 \rightarrow \mathbf{R}^3$:

$$\mathbf{x}(k) = \mathbf{f}_R(\mathbf{x}(k-1), \mathbf{u}(k)) \tag{4.13}$$

Donde

$$\mathbf{x}(k) = \begin{bmatrix} x(k) & y(k) & \theta(k) \end{bmatrix}^T \tag{4.14}$$

Es el vector de estado del sistema que define la posición del robot y

$$\mathbf{u}(k) = \begin{bmatrix} V(k) & \Delta\theta(k) \end{bmatrix}^T \tag{4.15}$$

Es el vector de entrada al sistema, por el cual se representa el desplazamiento del robot en un intervalo de tiempo. V representa el desplazamiento lineal del robot y $\Delta\theta_k$ su desplazamiento angular.

Debido a la inexactitud de las medidas del robot que intervienen de manera decisiva en la medición del desplazamiento, así como la discretización del movimiento de las ruedas se cometen errores en la medición del vector de entrada al sistema, por lo que se opta por modelarlo como

una variable aleatoria multidimensional gaussiana cuya media es el vector medido y su varianza justificará los errores cometidos en la medición del desplazamiento del robot. Suponiendo que los errores en los desplazamientos lineal y angular del robot en un intervalo de tiempo son independientes entre sí, la matriz de covarianza del vector de entrada al sistema será:

$$\mathbf{Q}(k) = \begin{bmatrix} \sigma_V^2 & 0 \\ 0 & \sigma_\theta^2 \end{bmatrix} \tag{4.16}$$

Donde σ_V^2 representa la covarianza del error cometido en la medición del desplazamiento lineal y σ_θ^2 representa la covarianza del error cometido en la medición del desplazamiento angular.

La obtención del valor de estos dos parámetros se realiza a partir de la repetición de dos experimentos.

El primer experimento consiste en realizar un desplazamiento lineal prefijado de un metro a una velocidad constante. Se calcula el desplazamiento entre muestreo y muestreo a partir de los encoders y se guarda su valor en un vector de medidas. De ese vector se puede extraer la media y la varianza suponiendo que la medida se ajusta a una distribución gaussiana.

El segundo experimento consiste en realizar un desplazamiento angular sobre el eje del robot a una velocidad constante. El proceso es el mismo, variando el hecho de que en vez de medir desplazamientos se mide el ángulo de giro del robot.

De estos dos experimentos se obtiene la matriz de covarianza dispuesta en (4.16) en función de los valores de σ_V^2 obtenido en el primer experimento y σ_θ^2 a partir del segundo experimento planteado.

4.1.3. Sistema sensorial del robot

El robot utilizado cuenta con distintos aparatos sensoriales, como se ha comentado con anterioridad. Para realizar el proyecto sólo son de utilidad las dos webcams que dispone. A continuación se describen los procesos a seguir para poder utilizar esas dos webcams como un sistema de medición de puntos característicos.

4.1.3.1. Modelo de cámaras y calibración

El calibrado de las cámaras es uno de los puntos clave a la hora de utilizar la visión como método de medida. A través del calibrado de una cámara se obtiene un modelo matemático a partir del cual se puede deducir la transformación de la porción del entorno vista por una cámara en la imagen en el plano bidimensional que proporciona como salida dicha cámara. Para obtener este modelo matemático se supone que las cámaras de las que se disponen son cámaras "pin-hole".

El modelo básico de la cámara "pin-hole" consiste en un centro óptico O, en donde convergen todos los rayos de la proyección, y un plano de imagen en el cual la imagen es proyectada. El plano de imagen está ubicado a una distancia focal f del centro óptico y perpendicular al eje óptico X. Este plano de imagen es paralelo a los ejes X y Y. Se puede definir también el punto R en la intersección del eje óptico y el plano de imagen, llamado punto principal o centro de imagen. En la figura 4.5 se representa geométricamente el plano imagen de una cámara y la proyección de un punto tridimensional.

Se puede definir un sistema de coordenadas bidimensional en el plano imagen con origen en la

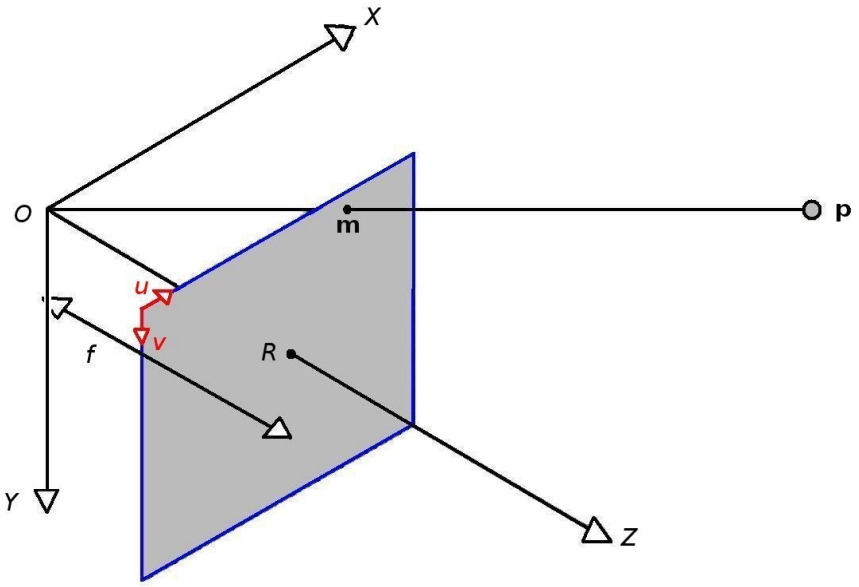

Figura 4.5: Modelo de cámara "pin-hole"

esquina superior derecha del plano de imagen y con ejes u y v, paralelos a X e Y respectivamente. Las coordenadas del punto $\mathbf{m}$ en el plano imagen serán relativas a este sistema de coordenadas.

Un punto $\mathbf{p} = (x, y, z)^\top$ en el espacio euclídeo de tres dimensiones se proyecta en el plano imagen en el punto $\mathbf{m} = (u, v)^\top$. El punto $\mathbf{m}$ en la imagen se define como la intersección entre la recta que une el centro óptico O y el punto $\mathbf{p}$ con el plano de imagen.

La proyección del punto $\mathbf{p}$ en la cámara se puede obtener mediante la siguiente ecuación en coordenadas homogéneas:

$$\begin{pmatrix} \lambda\mathbf{p} \\ \lambda \end{pmatrix} = \mathbf{P} \begin{pmatrix} \mathbf{p} \\ 1 \end{pmatrix} \tag{4.17}$$

donde $\mathbf{P}$ es una matriz 3×4 que se denomina matriz de proyección de la cámara. Dentro de dicha matriz están contenidos todos los parámetros necesarios para definir la proyección del punto $\mathbf{p}$ en la cámara.

La matriz de proyección $\mathbf{P}$ se puede expresar mediante la matriz parámetros intrínsecos y las matrices de parámetros extrínsecos:

$$\mathbf{P} = \mathbf{K} \begin{pmatrix} \mathbf{R} & \mathbf{T} \end{pmatrix}, \tag{4.18}$$

donde la matriz de parámetros intrínsecos $\mathbf{K}$ posee la información de las propiedades físicas de la cámara, tales como el tamaño de cada pixel o la distancia del punto focal a la matriz de sensores de colores. De forma simplificada la matriz $\mathbf{K}$ se define a partir de tres parámetros principales:

$$\mathbf{K} = \begin{pmatrix} f & 0 & d_u \\ 0 & f & d_v \\ 0 & 0 & 1 \end{pmatrix}, \tag{4.19}$$

La matriz de parámetros extrínsecos transforma a un origen de coordenadas conocido el punto focal de la cámara. Por ello se compone de una matriz de rotación $\mathbf{R}$ y un vector de desplazamiento $\mathbf{T}$.

Tanto los parámetros extrínsecos como intrínsecos deben ser calculados en un entorno real a partir de un método de calibración de los mismos. Para calcular los parámetros intrínsecos se ha partido de un patrón cuadriculado del que se conocen las dimensiones de sus cuadrículas y se ha utilizado un algoritmo de calibración estándar utilizado por las funciones de la librería OpenCV para calibrado de cámaras. En la figura 4.6 se puede ver el patrón visto por una de las cámaras utilizadas en el proyecto.

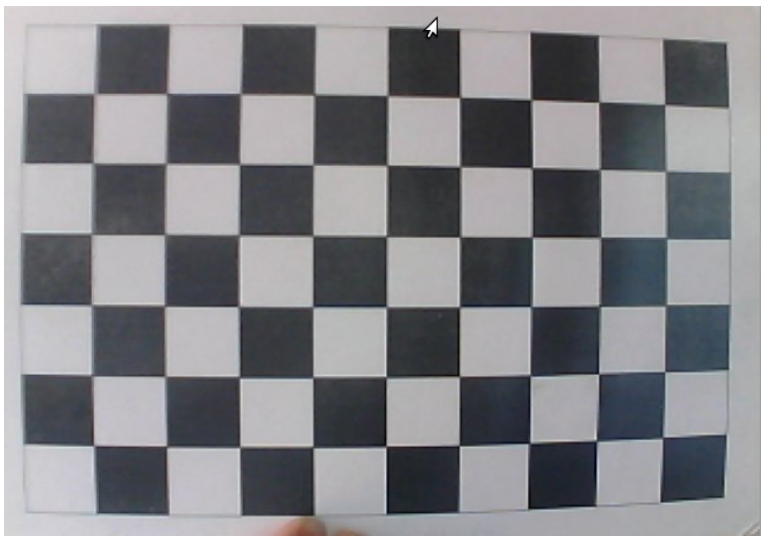

Figura 4.6: Patrón usado para la calibración de las cámaras

El modelo "pin-hole" utilizado no es preciso cuando la óptica de la cámara produce efectos no lineales en la forma en la que la luz se proyecta en el plano imagen. Para ello es necesario incluir un modelado explícito de dicha distorsión que se conoce como modelo polinómico de distorsión radial y tangencial. Mediante las funciones de calibración utilizadas es posible calcular los parámetros de distorsión radial y tangencial para una cámara, lo que permite rectificar las imágenes de modo que el modelo "pin-hole" sea preciso. En la figura 4.7 se puede ver la diferencia entre el patrón visto por la misma cámara sin haber corregido la distorsión y habiéndola corregido.

Visión estéreo El sistema propuesto consta de un conjunto esteroscópico formado por dos cámaras, que denominaremos cámara derecha e izquierda. Una vez se han calibrado las dos cámaras usadas en el proyecto por separado, es necesario obtener los parámetros extrínsecos que las asocian a ambas con los que poder relacionar los puntos entre el espacio tridimensional y las mediciones en las dos cámaras. Para obtener estos parámetros hay que apoyarse en la teoría de la geometría epipolar.

Sea $\mathbf{p}$ un punto en el espacio tridimensional, y $\mathbf{m}_i$ y $\mathbf{m}_d$ sus proyecciones en las dos imágenes correspondientes a la cámara izquierda y derecha respectivamente. La geometría epipolar explica

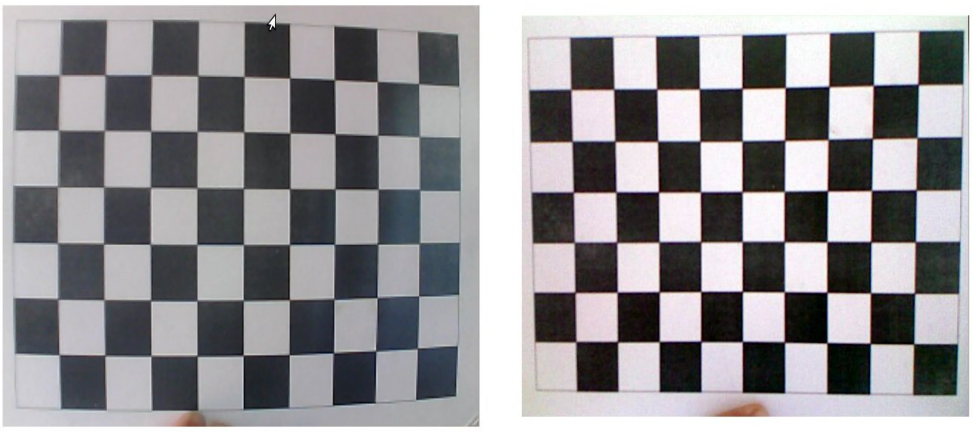

Figura 4.7: Izquierda: Imagen con distorsión; Derecha: Distorsión corregida

las relaciones entre $\mathbf{m}_i$ y $\mathbf{m}_d$. La figura 4.8 representa un esquema del caso de un punto en el espacio tridimensional visto por las dos cámaras.

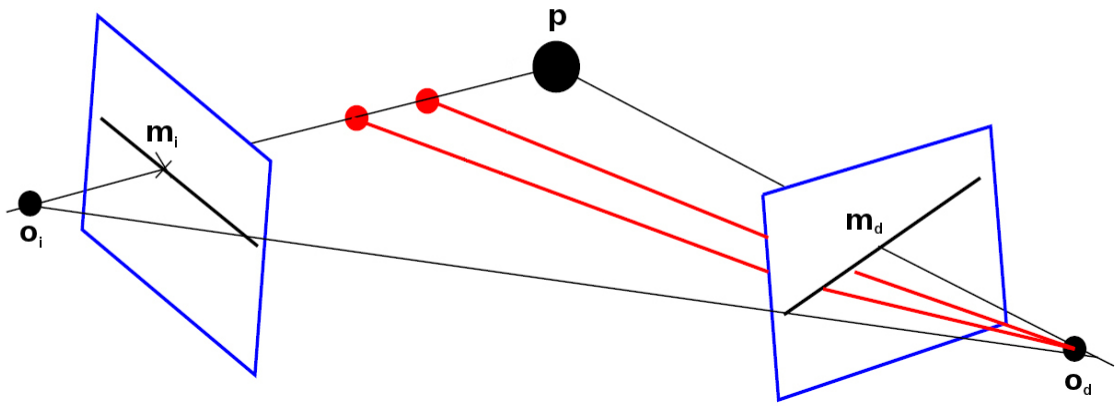

Figura 4.8: Representación de un sistema de visión estereoscópica

Como se puede ver en dicha figura, los centros ópticos $\mathbf{o}_i$, $\mathbf{o}_d$ de ambas cámaras, el punto $\mathbf{p}$ y sus proyecciones $\mathbf{m}_i$ y $\mathbf{m}_d$ son coplanares. Dados $\mathbf{o}_i$, $\mathbf{o}_d$ y $\mathbf{m}_i$, el punto correspondiente $\mathbf{m}_d$ tiene que pertenecer al plano formado por $\mathbf{o}_i$, $\mathbf{o}_d$ y $\mathbf{p}$. Por tanto, dado $\mathbf{m}_i$, el conjunto de puntos en el plano imagen de la cámara derecha que podrían ser correspondientes a $\mathbf{m}_i$ son aquellos que pertenecen a la recta formada por la intersección del plano $\mathbf{o}_i\mathbf{o}'_d\mathbf{m}_i$ y el plano imagen de la cámara con centro en $\mathbf{o}_d$. Esta linea recta se llama línea epipolar. El plano $\mathbf{o}_i\mathbf{o}_d\mathbf{m}_i$ se llama plano epipolar. Todos los planos epipolares intersectan en la línea $\mathbf{o}_i\mathbf{o}_d$, con lo que se deduce que todas las líneas epipolares tienen un punto de intersección común en cada plano imagen correspondiente. Estos puntos de intersección $\mathbf{e}_i$ y $\mathbf{e}_d$, denominados epipolos y corresponden con la intersección del vector $\mathbf{o}_i\mathbf{o}_d$ con el plano imagen de cada cámara. Dado un sistema estéreo determinado, los epipolos son únicos.

A partir de estas restricciones se puede calcular la relación entre las dos cámaras como una matriz de rotación y otra de traslación que definen la diferencia de separación y orientación entre las dos cámaras.

Para simplificar el problema de relacionar un punto proyectado en las dos imágenes se puede realizar una rectificación a las imágenes que consta de una transformación afín en cada imagen de cada cámara para que de manera virtual los planos imagen de las dos cámaras coincidan. Esto provoca que las líneas epipolares sean paralelas. Junto con esta transformación afín se realiza una corrección de la distorsión de cada cámara. Como resultado se obtienen dos matrices de proyección para las imágenes ya rectificadas.

La matriz de proyección para cada una de las cámaras, $\mathbf{P}_i$ para la cámara izquierda y $\mathbf{P}_d$ para la cámara derecha del sistema de visión estereoscópica, tendrán la siguiente forma:

$$\mathbf{P}_i = \begin{bmatrix} f & 0 & d_u & 0 \\ 0 & f & d_v & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$
$$\mathbf{P}_d = \begin{bmatrix} f & 0 & d_u & f * T_x \\ 0 & f & d_v & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{4.20}$$

4.1.3.2. Sistema de visión de marcas naturales

El sistema de visión de marcas naturales trata de transformar las dos imágenes vistas por las cámaras en un conjunto de puntos en un espacio tridimensional que correspondan con zonas de las imágenes con ciertas propiedades que las hacen interesantes para ser incluidas en el mapa del entorno. Estas propiedades tienen que ver con la posibilidad de que se pueda obtener un descriptor de estas zonas invariante a escala u orientación, haciendo posible detectar el mismo punto desde diferentes posiciones del robot. Los pasos a realizar para llegar a esta nube de puntos desde la captura de las dos imágenes se describen a continuación.

4.1.3.3. Deducir la posición en el espacio de las marcas

Sean dos cámaras "pin-hole" con matrices de proyección como las mostradas en (4.20) y orígenes de proyección $\mathbf{o}_i$ y $\mathbf{o}_d$ respectivamente. Sean las coordenadas de un punto en la imagen de cualquiera de las cámaras, por ejemplo $\mathbf{m}_i$. Se puede concluir que todos los puntos pertenecientes a la recta que une $\mathbf{o}_i$ y el punto $\mathbf{p}$ tienen como proyección $\mathbf{m}_i$. Esta recta infinita se puede obtener a partir de la calibración de la cámara y el punto medido en el plano imagen $\mathbf{m}_i$.

De este modo, dados $\mathbf{m}_i$ y $\mathbf{m}_d$, proyecciones de un punto tridimensional común $\mathbf{p}$ en ambas cámaras, el punto de corte de las dos rectas de cada imagen será el que corresponde a las coordenadas del propio punto $\mathbf{p}$. Debido a los errores en los parámetros intrínsecos y extrínsecos cometidos en la calibración de las cámaras, así como en la proyección del mismo punto en las dos imágenes provocarán que estas dos rectas no se corten. El método para obtener la posición de la marca consistirá en calcular la mínima distancia entre las dos rectas y el punto donde ocurre. Se supondrá como posición del punto buscado el punto medio de la recta que corta las dos rectas anteriores por los puntos en los que se encuentra la mínima distancia entre éstas. Sabiendo las ecuaciones de las dos rectas que proyectan los puntos detectados en las imágenes, se puede representar cada una de estas rectas por un vector $\mathbf{v}_i$, $\mathbf{v}_d$ y un punto $\mathbf{p}_i$, $\mathbf{p}_d$. La recta buscada tendrá como vector director $\mathbf{v} = \mathbf{v}_i \times \mathbf{v}_d$ y pertenecerá a los planos formados por el vector $\mathbf{v}$ y los vectores $\mathbf{p}_i - \mathbf{o}_i$ y $\mathbf{p}_d - \mathbf{o}_d$, de modo que la intersección de estos dos planos dará como resultado la recta buscada. De dicha recta se encuentran los puntos $\mathbf{p}_a$ y $\mathbf{p}_b$ que tienen como proyección $\mathbf{m}_i$ y $\mathbf{m}_d$ respectivamente. La estimación de $\mathbf{p}$ en este caso se encuentra como

$\mathbf{p} = \frac{1}{2}(\mathbf{p}_a + \mathbf{p}_b)$. En la figura 4.9 se puede ver una representación del método implementado.

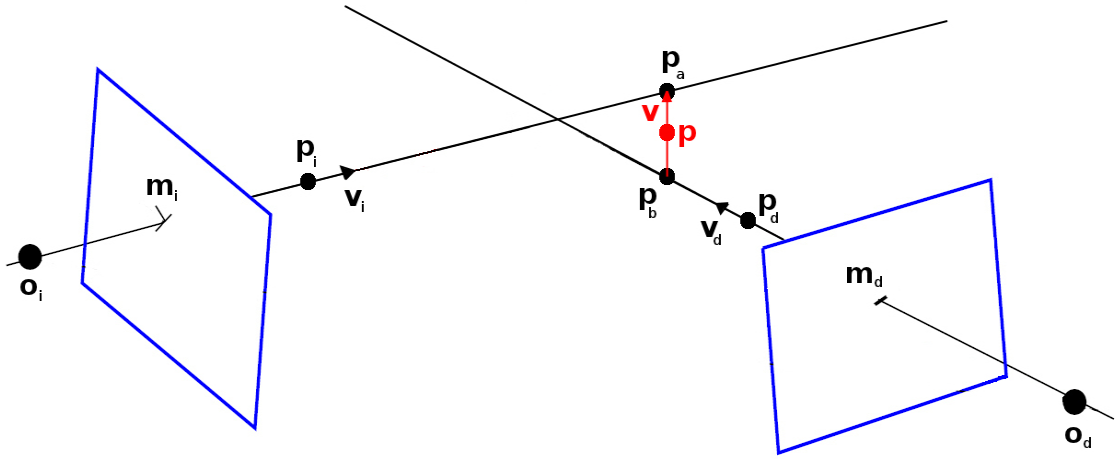

Figura 4.9: Representación de medición de puntos en el espacio a partir de dos cámaras

Una vez llegados a este paso se puede filtrar los posibles errores por asociaciones incorrectas de puntos en la imagen. En primer lugar la profundidad del punto no puede ser negativa. Para eliminar el resto de posibles falsas asociaciones se reproyecta el punto hallado a las dos imágenes. Las coordenadas obtenidas deben estar en el margen de pixels de las imágenes originales.

Se puede representar el punto tanto con sus coordenadas euclídeas como con su posición en una de las imágenes y la disparidad con respecto a la otra imagen. Este otro método de representar el punto permitirá visualizar de una manera más intuitiva el funcionamiento del nodo de asociación de datos.

4.1.3.4. Selección de las marcas naturales en las cámaras

Existen en la literatura varios algoritmos que seleccionan marcas de una imagen, generalmente marcas que pueden ser descritas como esquinas, que cumplen una cierta clase de características. La librería OpenCV dispone de una clase llamada FeatureDetector desde la que implementa varios detectores, cada uno haciendo uso de un algoritmo diferente, pero todos ellos compartiendo los mismos métodos, parámetros de entrada y de salida.

OpenCV implementa los algoritmos FAST [42], Good Features to Track [43], MSER [44], CenSurE [45], SIFT [46] y SURF [47]. En este proyecto se ha elegido el algoritmo "Good Features to Track" propuesto originalmente por Jiambo Shi y Carlo Tomasi.

Este algoritmo no extrae marcas invariantes a escala ya que no detecta puntos sobre múltiples niveles de una pirámide gaussiana como hace por ejemplo SURF, pero la estabilidad a la hora de encontrar siempre los mismos puntos en la imagen y el menor tiempo empleado por el algoritmo en realizar el procesado en la imagen le hacen un algoritmo interesante teniendo en cuenta que el objetivo final es realizar un mapa del marcas del entorno según el robot va avanzando.

Para poder relacionar una marca en las imágenes de las dos cámaras hace falta encontrar alguna propiedad en común que relacione una marca vista en una cámara, otra marca en la otra cámara y sea capaz de discernir si pertenecen a la misma marca en el espacio tridimensional. Como ya se ha visto en la explicación del sistema de visión estereoscópica, para que dos puntos vistos en cada una de las cámaras pertenezcan al mismo punto en el espacio tridimensional,

deben estar incluidos en sus líneas epipolares. El problema es que este requisito es necesario pero no único. Para poder asociar dos marcas vistas por dos cámaras diferentes se puede extraer un descriptor de cada marca y compararlos. Suponiendo que una marca es un parche de un determinado número de pixels en la imagen, centrado en la posición en la que se proyecta el punto del que queremos extraer su descriptor, el descriptor se obtiene a través de diversos algoritmos que relacionan los puntos incluidos en dicho parche para generar un código o descriptor que define al punto. Algunos de estos algoritmos, como en el caso de la extracción de marcas, están ya incluidos en la librería OpenCV.

OpenCV implementa los descriptores de marcas SIFT, SURF y BRIEF [48]. Se ha optado por utilizar descriptores BRIEF. Estos descriptores no son invariantes a escala ni a rotaciones, pero asumiendo que el detector de marcas no tiene en cuenta la escalabilidad de los puntos y que al permanecer el robot del proyecto con la cámara fija no habrá rotaciones, le hace suficiente para cumplir los requisitos, añadiendo menos tiempo de procesado que algoritmos como SIFT y SURF.

Una vez extraídos los descriptores de una imagen, para relacionarlos con los extraídos en la cámara opuesta se calcula la distancia euclídea entre los dos descriptores. De este modo para asociar dos puntos de un mismo objeto en el espacio tridimensional se buscan aquellos con menor distancia euclídea entre descriptores y que a su vez cumpla que los puntos en las dos cámaras pertenezcan a su línea epipolar.

La búsqueda de marcas naturales en una imagen no produce siempre los mismos puntos, con lo que el resultado final del algoritmo de búsqueda y asociación de puntos naturales provocará que un mismo punto se vea intermitentemente en el tiempo aun a pesar de no mover el robot. Para evitar este efecto se propone realizar un seguimiento de los puntos ya asociados en las imágenes de tiempos posteriores. Bruce D. Lucas y Takeo Kanade propone un algoritmo a tal efecto, [49]. El algoritmo conocido por método Lucas-Kanade es ámpliamente utilizado para la estimación del flujo óptico de dos imágenes. Asume que el flujo es constante en una región cercana al pixel en consideración y resuelve las ecuaciones del flujo óptico para los pixels de su alrededor por el criterio de mínimos cuadrados.

4.1.3.5. Modelado de la posición de las marcas

Se puede modelar la posición de una marca en el espacio como una distribución normal en la que la media será la posición medida de la marca por el método anteriormente mencionado. Para calcular la varianza de la medida hay que tener en cuenta cuanto error se puede estar cometiendo al realizar la medida. Considerando que las coordenadas son variables aleatorias independientes, la varianza de la medida será una matriz con elementos no nulos en la diagonal principal. Suponiendo una desviación típica de 3 pixels en cada coordenada, dará como resultado una matriz de covarianza $\mathbf{P} = \mathbf{I} * 9$, donde $\mathbf{I}$ es la matriz diagonal.

$$\mathbf{P} = \mathbf{I} * \sigma_l^2 = \begin{bmatrix} 9 & & \\ & 9 & \\ & & 9 \end{bmatrix} \tag{4.21}$$

4.1.3.6. Modelo de medición de marcas desde el robot

El modelo de medición de marcas trata de obtener la relación entre la medida de una marca en el par estéreo, la posición del robot y la posición en el espacio de dicha marca.

La función de observación representará las marcas añadidas al mapa tal como las deberían ver las cámaras del robot en función de la posición almacenada de dichas marcas y en función también de la posición del robot. La función de observación indicará por lo tanto los pixels en la imagen en la que debería aparecer cada una de las marcas en función del estado del sistema que se ha estimado en la etapa anterior.

Para evitar posibles confusiones en el desarrollo de las ecuaciones se omitirá la dependencia con el tiempo de las variables de estado del sistema.

El primer paso es realizar un cambio de coordenadas de cada marca pasando del origen de coordenadas global a un origen de coordenadas centrado en el robot. Este cambio del origen de coordenadas supone una rotación y una traslación de la marca:

$$\mathbf{h}_{c_i} = h_1(\mathbf{x}_R, \mathbf{x}_F) = \mathbf{R}_{rw} \cdot (\mathbf{x}_{F_i} - \mathbf{r}_{wr}) \tag{4.22}$$

Donde el vector $\mathbf{h}_c$ representa las coordenadas de una marca vista desde el robot:

$$\mathbf{h}_{c_i} = \begin{bmatrix} h_{c_{ix}} \\ h_{c_{iy}} \\ h_{c_{iz}} \end{bmatrix} \tag{4.23}$$

La matriz $\mathbf{R}_{rw}$ representa la rotación que sufren las marcas debido a la rotación del robot con respecto al eje de coordenadas global:

$$\mathbf{R}_{rw} = \begin{bmatrix} \cos(\theta_R) & \sin(\theta_R) & 0 \\ -\sin(\theta_R) & \cos(\theta_R) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.24}$$

El vector $\mathbf{r}_{wr}$ representa la traslación que sufren las marcas debido a la propia traslación del robot con respecto al eje de coordenadas global:

$$\mathbf{r}_{wr} = \begin{bmatrix} x_R \\ y_R \\ 0 \end{bmatrix} \tag{4.25}$$

Y el vector $\mathbf{x}_{F_i}$ representa la posición de la marca almacenada en el mapa. Esta posición se corresponde con el estado de la marca i de la matriz de estados.

$$\mathbf{x}_{F_i} = \begin{bmatrix} x_{F_i} \\ y_{F_i} \\ z_{F_i} \end{bmatrix} \tag{4.26}$$

Una vez se tienen la marca con su referencia de coordenadas situada en el robot, se puede deducir a partir del modelo de cámara pin-hole las coordenadas en pixels en las que se debería observar la marca con la cámara:

$$\mathbf{y}_i = h_2(\mathbf{h}_{c_i}) = \begin{bmatrix} u_0 - f_u \cdot \dfrac{h_{c_{iy}}}{h_{c_{ix}}} \\ v_0 - f_v \cdot \dfrac{h_{c_{iz}}}{h_{c_{ix}}} \end{bmatrix} \tag{4.27}$$

A partir de las anteriores ecuaciones se puede definir la función de observación de la siguiente manera:

$$\mathbf{y}_i = h(\mathbf{x}_R, \mathbf{x}_{F_i}) = h_2(h_1(\mathbf{x}_R, \mathbf{x}_{F_i}) \tag{4.28}$$

En el caso de que se deseara conocer la posición de la marca en un sistema estereoscópico, se define dicha medida como:

$$\mathbf{y}_{stereo_i} = \begin{bmatrix} \mathbf{y}_i \\ f_u \cdot \dfrac{T_x}{h_{c_{i_x}}} \end{bmatrix} \tag{4.29}$$

4.2. Visual SLAM para posicionamiento de robots

Para implementar el iSAM se va a hacer uso de la librería GTSAM que proporciona todos los Factor Graph típicos que modelan las relaciones entre diferentes posiciones de un robot y posiciones y medidas de cámaras, tanto estéreo como monoculares.

4.2.1. Modelado del sistema a partir de Factor Graphs

Un Factor Graph es un modelo gráfico utilizado para modelar problemas complejos de estimación, como es por ejemplo el SLAM. Un Factor Graph es un grafo bipartito consistente en factores conectados a variables. Las variables representan las variables aleatorias desconocidas en el problema de estimación, mientras que los factores representan la información probabilística de esas variables proveniente de las medidas o del conocimiento previo.

Para explicar gráficamente en que consiste el Factor Graph, se puede empezar por la representación de una red Bayesiana para un modelo de Markov oculto (HMM) sobre tres iteraciones de tiempo 4.10. En una red Bayesiana, cada nodo está asociado con una densidad de probabilidad condicional: La cadena superior engloba la probabilidad a priori $P(X_1)$ y las probabilidades condicionales $P(X_2|X_1)$ y $P(X_3|X_2)$, mientras que las medidas Z_t dependen únicamente del estado X_t, modeladas por densidades de probabilidad condicional $P(Z_t|X_t)$. Conociendo las me-

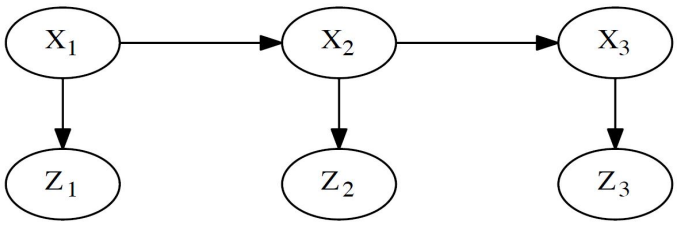

Figura 4.10: HMM sobre tres pasos representado como red Bayesiana

didas z_1, z_2 y z_3 interesa encontrar la secuencia de estados internos (X_1, X_2, X_3) que maximiza la probabilidad a posteriori $P(X_1, X_2, X_3|Z_1 = z_1, Z_2 = z_2, Z_3 = z_3)$. Como las medidas Z_1, Z_2 y Z_3 son *conocidas*, la probabilidad a posteriori es proporcional a seis factores, tres de los cuales derivan de la cadena de Markov y los otros tres son factores de similitud definidos como $L(X_t; z) \propto P(Z_t = z|X_t)$:

$$P(X_1, X_2, X_3|Z_1, Z_2, Z_3) \propto P(X_1)P(X_2|X_1)P(X_3|X_2)L(X_1; z_1)L(X_2; z_2)L(X_3; z_3) \tag{4.30}$$

Esto motiva un modelo gráfico diferente, que se llamará Factor Graph, en el que únicamente se representa las variables aleatorias que suponen una incógnita conectadas por factores que engloban la información probabilística dentro de ellos, como en la figura 4.11. Para buscar el máximo a posteriori, entonces, hay que maximizar el producto

$$f(X_1, X_2, X_3) = \prod f_i(\chi_i) \tag{4.31}$$

Osea, el valor del Factor Graph.

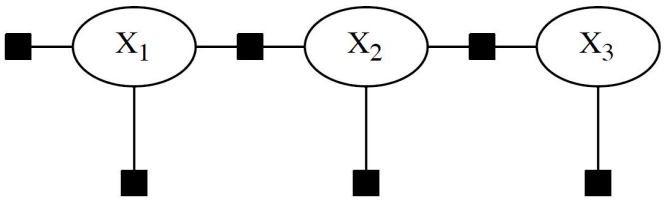

Figura 4.11: HMM con las medidas observadas representado como Factor Graph

Para solucionar el problema del Visual SLAM usando iSAM, hay que definir los factores necesarios en el modelo. Se usarán tres tipos de factores para definir el Factor Graph:

- Factor para modelar el movimiento del robot.
- Factor para modelar la adición de nuevas marcas al mapa.
- Factor para modelar la visualización de una marca ya añadida desde la posición actual del robot.

Factor para modelar el movimiento del robot

Las variables aleatorias de este factor serán cada una de las posiciones del robot que va tomando en la trayectoria que sigue representadas en la ecuación 4.2.

Su información probabilística es la relación entre la posición nueva y la posición anterior del robot. Esta información probabilística viene definida por el modelo de movimiento del robot 2.2. Es necesario indicar el ruido en la medida de odometría usado. En este caso se usa la siguiente matriz diagonal para representar el error de odometría:

$$\Lambda_i = \begin{bmatrix} 0{,}001^2 & \\ & 0{,}03^2 \end{bmatrix} \tag{4.32}$$

Donde el elemento de la posición $(0, 0)$ representa la varianza del error en el desplazamiento lineal medida en metros σ_V^2 y el elemento de la posición $(1, 1)$ representa la varianza del error en el desplazamiento angular del robot medida en radianes σ_θ^2, tal como se mencionan en 4.16.

Factor para modelar la adición de nuevas marcas al mapa

Las variables aleatorias en este factor serán cada una de las nuevas marcas del entorno detectadas que han de añadirse al mapa: 4.1 Su información probabilística relaciona la medida de las marcas del entorno en el sistema de visión estéreo con la posición del robot en el momento

de la medición de la nueva marca. Esta información probabilística viene definida por el modelo de medida del robot 4.29. Es necesario indicar el ruido en la medida estéreo de la marca usada. En este caso se usa la siguiente matriz diagonal para representar el error de la medida estéreo:

$$\Sigma_{stereo_k} = \begin{bmatrix} 3^2 & & \\ & 3^2 & \\ & & 3^2 \end{bmatrix} \tag{4.33}$$

Donde los elementos no nulos representan la varianza de cada una de las variables de la medida σ_l^2 en píxels 4.21.

Factor para modelar la detección de una marca

Las variables aleatorias en este factor serán las marcas del entorno detectadas y ya existentes en el mapa 4.1 Su información probabilística relaciona la medida de las marcas del entorno en el sistema de visión monocular con la posición del robot en el momento de la medición de la nueva marca. Se usa visión monocular en vez de estéreo cuando las marcas ya se han añadido porque, al estar ya inicializada la marca con su información estéreo, la incertidumbre que provoca el desconocimiento de la profundidad que tiene el hecho de usar una única cámara fue solucionado, permitiendo al proceso de asociación de datos que solo tenga que realizar operaciones sobre una única imagen del par estéreo hasta que sea necesario añadir nuevas marcas al mapa. Esta información probabilística viene definida por el modelo de medida del robot 4.28. Es necesario indicar el ruido en la medida estéreo de la marca usada. En este caso se usa la siguiente matriz diagonal para representar el error de la medida en la cámara:

$$\Sigma_{monocular_k} = \begin{bmatrix} 3^2 & \\ & 3^2 \end{bmatrix} \tag{4.34}$$

Donde los elementos no nulos representan la varianza de cada una de las variables de la medida σ_l^2 en píxels 4.21.

4.2.2. Ciclo del trabajo del algoritmo

Existen dos tipos de eventos que hacen reaccionar al sistema desarrollado:

- Captura de la escena.
- Desplazamiento del robot.

4.2.2.1. Captura de la escena

Con la llegada de un par estéreo desde el sistema sensorial se ejecuta la rutina de extracción y medición de marcas de entorno. Los pasos que se realizan en dicha rutina son los siguientes:

1. Extracción de marcas de la escena.
2. Comparación de escenas.
3. Adición de una nueva escena al vector de escenas.

4. Corrección de índices de marcas.
5. Adición de Factor Graphs para marcas nuevas.
6. Adición de Factor Graphs para marcas detectadas.

Extracción de marcas de la escena En la etapa de extracción de marcas solo se usa una de las dos cámaras del robot y se mide la posición de las marcas como el lugar en pixels donde se observa cada una de ellas en la imagen.

Comparación de escenas Una asociación de datos incorrecta lleva rápidamente a la divergencia del mapa, con lo que es vital cometer la mínima cantidad de errores al asociar los puntos del mapa con los medidos.

Una de las estrategias más utilizadas para la asociación de puntos entre el mapa y la observación de la escena es la de emparejar en función de la distancia entre objetos, asociando siempre el objeto medido con el objeto del mapa más cercano. Esta estrategia se llama "Nearest Neighbour"

La principal ventaja de esta estrategia de emparejamiento es su simplicidad, pero también provoca varios inconvenientes. El mayor inconveniente es que al considerar cada emparejamiento de manera independiente no usa la relación existente entre distintas observaciones para añadir mayor robustez a la etapa de asociación de datos.

Este inconveniente se puede mitigar aplicando el test de compatibilidad conjunta o "Joint Compatibility". Esta estrategia se basa en el cálculo de la distancia conjunta de Mahalanobis para un un conjunto de emparejamientos mediante el vector acumulado de sus innovaciones y la matriz conjunta de varianzas y covarianzas de dichas innovaciones, para establecer la posibilidad de que todos los emparejamientos considerados puedan ser compatibles simultneamente. Se suelen calcular primero los emparejamientos que cumplen el test de compatibilidad simple para luego realizar la prueba de compatibilidad conjunta.

Otro método para evitar malas asociaciones de objetos es aplicando el algoritmo RANSAC, para que a partir de la obtención de la matriz de transformación que relaciona dos escenas a partir de un grupo de asociaciones previo, pueda discriminar aquellas asociaciones que se separan de la transformación definida.

Como se ha mencionado anteriormente, el mayor esfuerzo de esta etapa irá dirigido a evitar errores en la asociación de las marcas vistas por el robot con las almacenadas previamente. Hay que tener en cuenta también que un algoritmo que descarte la mayoría de las marcas por no estar seguro de que sean válidas también es malo, pues no corregirá nunca el estado del mapa y ocurrirá el mismo problema de divergencia que si se realizan falsas asociaciones.

La asociación se realiza en dos etapas. La primera de ellas consiste en un test de compatibilidad individual, en el que se asocian aquellas marcas vistas por el robot cuya distancia euclídea del descriptor BRIEF a su par de la escena almacenada sobre la que se está haciendo la asociación es la menor. Tras esta etapa se tendrá un conjunto de pares marcados como posibles candidatos.

La segunda etapa es un test de compatibilidad conjunta. Este test trata de contemplar el hecho de que un error en la estimación de la posición del robot afectará por igual a todas las marcas, con lo que aquellas marcas que tengan errores en la medida de magnitudes o direcciones diferentes a la media, con mucha probabilidad, serán falsos positivos que hay que filtrar. Para realizar este test se ha implementado una versión del algoritmo RANSAC.

Para seleccionar la escena que corresponde a la vista actual se calcula el porcentaje de asociaciones que han pasado el test de compatibilidad conjunta con respecto al total de asociaciones y se selecciona aquella con mejor resultado a no ser que esté por debajo de la mitad, en cuyo caso se añadirá la imagen como nueva a un vector de escenas que contiene las escenas que mejor describen el trayecto realizado por el robot.

Una vez está detectada la escena vista, la asociación de puntos del mapa con los vistos en la escena ya está realizada, pues cada una de las imágenes almacenadas en el vector de escenas también lleva asociada los puntos del mapa que se vieron en esa imagen. Estas asociaciones permitirán cerrar lazos en el recorrido del robot, conteniendo el crecimiento del error que, sin esta solución, sería difícil de gestionar.

Adición de una nueva escena al vector de escenas En caso de que no se considere ninguna escena como detectada se extraen las marcas de la imagen derecha de la escena actual. En caso de que no se considere ninguna escena como detectada se hace uso de la imagen de disparidad que provee el sistema de visión estéreo.

Aquellos puntos de la escena actual que no se han podido asociar en la mejor escena seleccionada son candidatos a ser nuevos puntos que incluir al mapa. Se busca la información de disparidad medida de cada uno de los puntos para obtener la información estéreo necesaria para inicializarlo.

Aparte, la escena se aade al grupo de escenas caractersticas del trayecto del robot.

Corrección de índices de marcas Cada marca en las escenas almacenadas tiene asociado un identificador único. Puede darse el caso de que se añada una marca ya almacenada en el mapa y pasado un tiempo, el sistema de asociación de escenas detecte esa marca en una escena anterior con un índice diferente. Para que el sistema sea coherente se debe modificar uno de los dos índices en los factores añadidos al grafo, así como eliminar uno de los dos valores usados en la inicialización del punto. Para ello se crea una asociación entre el índice a cambiar y el índice que se debe usar cuando se detecta la situación descrita. Cuando se detecta este tipo de asociación se procede a la modificación del grafo y del vector de inicialización de variables.

Adición de factores al Factor Graph para marcas nuevas Se recorren todas las marcas detectadas como nuevas en la etapa de adición de la escena actual al grupo de escenas características. Cada una de estas mediciones se añaden al iSAM creando los factores tal como se ha indicado en 4.2.1

Adición de factores al Factor Graph para marcas detectadas Se recorren todas las marcas detectadas como asociadas en la etapa de comparación de la escena actual con el grupo de escenas características. Cada una de estas mediciones se añaden al iSAM creando los factores tal como se ha indicado en 4.2.1

4.2.2.2. Desplazamiento del robot

Cada vez que llega un mensaje de odometría hace falta ejecutar la rutina que actualiza la posición del robot. El mensaje de odometría que entrega el sistema de localización del robot representa la posición estimada del robot en el instante de tiempo en que ha llegado dicho mensaje. De este modo primero se calcula el desplazamiento entre la posición anterior y la nueva

posición para calcular la odometría que hay que introducir al Factor Graph que relaciona las dos posiciones mencionadas. Una vez añadida la odometría junto con su error y la posición que llega del mensaje como valor inicial para la nueva posición, se realiza una etapa de actualización del iSAM. Se realiza la etapa de actualización tras añadir cada nueva posición para asegurar que la última variable aleatoria en el grafo sea una posición, tal como se indica en las consideraciones expuestas en los fundamentos teóricos del iSAM.

4.3. Implementación del algoritmo

El algoritmo se ha desarrollado en lenguaje C++, con el apoyo del entorno de programacin Eclipse y las librerías y nodos proporcionadas por la plataforma de desarrollo ROS así como las librerías GTSAM, específicas para el uso de *Smoothing and Mapping*, sobre un sistema operativo con la distribución Ubuntu. El algoritmo se ha desarrollado como varios nodos de ROS, con lo que hace falta que el núcleo del sistema ROS esté en funcionamiento.

4.3.1. ROS

ROS es un meta-sistema operativo que proporciona servicios como abstracción del hardware, comunicación entre procesos mediante mensajes y manipulación de paquetes. ROS dispone de una multitud de nodos ya realizados que dota de una gran funcionalidad al sistema completo.

Instalación de ROS

ROS está soportado en la distribución Ubuntu de Linux, y se encuentra en fase experimental en el resto de distribuciones de Linux, así como en OS X y Windows.

Para instalar ROS en Ubuntu hay que seguir una serie de pasos que se detallan a continuación:

1. Configurar los repositorios de Ubuntu para permitir “restricted”, “universe”, y “multiverse”.
2. Configurar el sistema para aceptar software desde “packages.ros.org”.
3. Añadir las claves del repositorio de ROS.
4. Instalar el paquete “ros-indigo-desktop-full” con “apt-get”.

Comunicación entre procesos

La comunicación entre procesos o nodos es uno de los puntos fuertes que pone ROS a disposición de los desarrolladores. Existen tres maneras de comunicar procesos. La primera de ellas es el método clásico de cliente-servidor, en el que un nodo hace de servidor y varios clientes se comunican con él. El segundo método es almacenando datos en un servidor de parámetros que comparten todos los nodos. El tercer método es mediante mensajes. Este método se consigue creando una tubería llamada *topic* en la que el nodo que quiere enviar un mensaje lo publica, mientras que hay uno o varios nodos que pueden suscribirse a ese *topic* y recibirán el mensaje. Cuando se envía un mensaje al *topic*, en todos aquellos nodos que estén escuchando el *topic* se generará un callback para poder recibir el mensaje y actuar en consecuencia. Este método de envío de mensajes es el utilizado en el algoritmo desarrollado.

Nodos existentes utilizados

Como se ha indicado con anterioridad existen numerosos nodos ya realizados para ROS. Para el caso particular de la visión estereoscópica existen ya nodos de calibración de cámaras estéreo y rectificación de sus imágenes. Se debe arrancar el nodo de visión estéreo para poder empezar a recibir imágenes de las dos cámaras así como los parámetros de calibración de éstas. Las imágenes se publicarán en un *topic* diferente según sea la cámara de la que vengan. De manera análoga, existirá un *topic* para los parámetros de calibración de las cámaras.

Otro de los nodos que se utilizan es el de odometría. Este nodo se encarga de publicar el desplazamiento lineal y angular que se ha medido en el robot. El desplazamiento se envía a través del *topic* de odometría.

Otro nodo utilizado es el que indica la posición de las cámaras del robot con respecto a su base. Este nodo define una rotación y una traslación para poder relacionar la base de las cámaras con la base de movimiento del robot.

Se han creado dos nodos que se suscriben a los *topics* de los nodos anteriormente mencionados. Estos nodos son los encargados de realizar la etapa de asociación de datos y la gestión del mapa.

RVIZ

RVIZ es un nodo de ROS para visualización de entornos en 3D. Este nodo puede ser configurado para suscribirse a diferentes *topics* relacionados con posicionamiento de objetos en espacios tridimensionales. En el caso del algoritmo desarrollado, se crea un *topic* para enviar la posición de las marcas almacenadas en el mapa, aparte de dos transformaciones, también llamadas TF en ROS. Las marcas almacenadas se envían como un array de puntos tridimensionales, la posición medida por la propia odometría del robot sin corrección se envía como una transformación TF del eje de coordenadas desde el centro de coordenadas global hasta la posición estimada del robot haciendo uso solo de la odometría. La posición corregida del robot por el algoritmo se representa como otra transformación TF desde la posición estimada por la odometría hasta la posición corregida por el propio algoritmo SLAM.

4.3.2. Organización del algoritmo SLAM

El algoritmo se ha organizado en varios paquetes en función de la parte del algoritmo que solucionan.

El paquete principal es el paquete visual_isam. Este paquete se encarga de la implementación práctica del algoritmo iSAM haciendo uso de las librerías GTSAM.

El paquete points_publisher se encarga de transformar la información entregada por los nodos del paquete visual_isam y dibujarlo como una nube de puntos para poderla visualizar en RVIZ.

El paquete scenes_detection se encarga del apartado específico de asociación de datos.

Diagrama de bloques del sistema propuesto

En la figura 4.13 se muestra el diagrama de bloques de la propuesta, destacando los procesos y funciones que componen el sistema de posicionamiento simultáneamente a la construcción de un mapa del entorno.

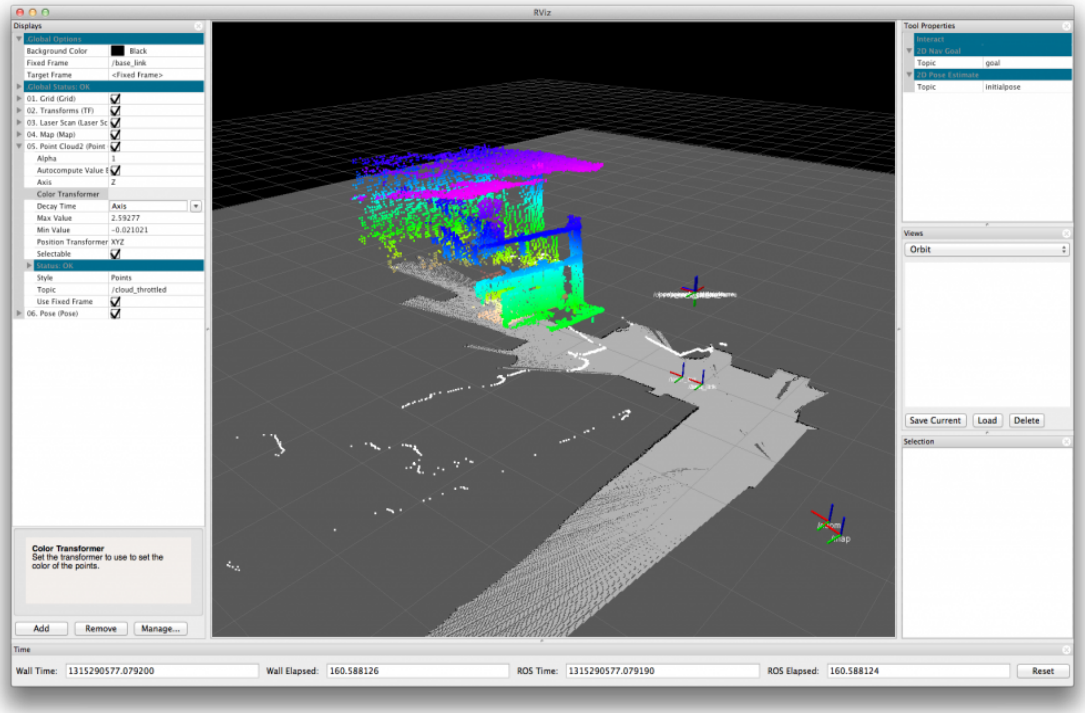

Figura 4.12: Nodo RVIZ de ROS

Paquete visual_isam

Dentro de este paquete se pueden encontrar dos nodos ejecutables con las misma funcionalidad pero un enfoque diferente:

- El primer nodo es el nodo visual_iSAM, que implementa la solución principal al SLAM, tal como se ha descrito en los apartados anteriores. Este nodo ya ejecuta el algoritmo online.
- El segundo nodo es el nodo visual_iSAM2. Este nodo hace uso del algoritmo propuesto por [23], siendo una mejora al algoritmo iSAM. Este nodo aun no es funcional, se encuentra en desarrollo.

Señales de entrada al nodo de SLAM El nodo de SLAM se subscribe a los siguientes mensajes para realizar el ciclo de trabajo:

- Mensaje de odometría. Este mensaje debe haber sido publicado por la plataforma robótica e indicar la posición del robot relativa a un origen de coordenadas global.
- Mensaje de marcas. Este mensaje se publica por parte del nodo de asociación de datos. Su información son tres vectores, uno indicando las marcas que hay que fusionar sobre el mismo índice, otro con la posición de las nuevas marcas a añadir al mapa y el último con la posición de las marcas detectadas del mapa.
- Parámetros de calibración del sistema estéreo. Este mensaje es necesario para crear los Factor Graph de la medición de las marcas con el uso de las librerías GTSAM.

El nodo de SLAM publica los siguientes mensajes como resultado del algoritmo:

- Pose del robot original. Esta es la posición actual del robot en cada instante de tiempo sin hacer uso del algoritmo.
- Marcas en el espacio sin realizar correcciones. Se guarda la posición en el espacio con la que se inicializa cada marca del mapa y se publica para poder compararla con la posición que entregará el algoritmo de SLAM.
- Camino estimado que ha recorrido por el robot sin realizar correcciones a la trayectoria.
- Pose del robot corregida. Esta es la posición actual del robot en cada instante de tiempo que da como solución el algoritmo implementado.
- Marcas en el espacio corregidas. La posición de las marcas se actualizará por el algoritmo cada vez que se ejecute la fase de actualización.
- Camino estimado que ha recorrido por el robot sin realizar correcciones a la trayectoria.

Estos mensajes se publican cada vez que se realiza una fase de actualización del SLAM.

Se define un servidor en el nodo de SLAM para obtener el índice de la posición actual del robot, necesario por parte del nodo de asociación de datos para asignar una posición del robot a las medidas de las marcas.

Con la llegada de cada uno de los mensajes a los que está suscrito el nodo se ejecuta la función encargada a realizar las tareas asociadas a dicho mensaje.

Mensaje de odometría Con la llegada de un mensaje de odometría se añade una nueva posición al Factor Graph correspondiente a la posición actual del robot. El mensaje de odometría se corresponde a una posición sin corrección de los errores acumulados. Para calcular la posición correctamente hace falta almacenar la posición anterior del robot sin corrección para calcular el desplazamiento del robot entre las dos posiciones y aplicar dicho desplazamiento a la última posición conocida y corregida del robot. El resultado de añadir el desplazamiento calculado a la última posición es la posición nueva que hay que añadir al Factor Graph.

Tras la adición de la nueva posición se realiza una actualización del iSAM.

Mensaje de marcas El mensaje de marcas está compuesto por tres vectores:

Vector de corrección de índices

Los elementos del vector indican si hay algún par de marcas que deben ser unidas bajo un mismo índice, ya que representan al mismo punto del mapa. Cuando el vector no se encuentra vacío se ejecuta el proceso de corrección de índices, que elimina la inicialización de la marca a la que se desea fusionar y se cambia el índice en todas aquellas asociaciones que se encuentran en el grafo.

Vector de adición de nuevos puntos

Los elementos del vector indican si hay algún punto nuevo a añadir al mapa de marcas. Cuando el vector no se encuentra vacío se ejecuta el proceso de adición de nuevas marcas. Cada punto se define como su posición en la imagen rectificada del sistema estereoscópico y su disparidad, extraida del mapa de disparidad asociado a la imagen rectificada. Con estos datos y la información de los parámetros intrínsecos y extrínsecos del sistema estéreo se crea el factor para el nuevo punto y su valor en el espacio para realizar la inicialización.

Vector de medidas de marcas

Los elementos del vector indican si hay alguna marca ya existente en el mapa que ha sido detectada en la escena actual. Cuando el vector no se encuentra vacío se ejecuta el proceso de creación de nuevas mediciones del punto. Cada punto se define como su posición en la imagen rectificada del sistema estereoscópico sin necesidad de su información de disparidad. Los parámetros de calibración del sistema estéreo se deben usar de igual manera que para el caso de la adición de nuevos puntos al mapa.

Mensaje con los parámetros de calibración Con la llegada de este mensaje se actualizan los atributos internos en los que se almacena la información sobre los parámetros de calibración del sistema de visión estereoscópica.

4.3.2.1. Paquete scenes_detection

Dentro de este paquete se encuentra un único nodo ejecutable, llamado scenes_detector, encargado de realizar las tareas de asociación de datos para el SLAM.

Este nodo se subscribe a los siguientes mensajes:

- Mensaje de la cámara rectificada. Este mensaje debe haber sido publicado por la plataforma robtica tras una calibración del sistema estéreo que debe disponer. El mensaje es una imagen de la escena que está viendo el robot en el instante actual.

- Mensaje del mapa de disparidad. Este mensaje debe haber sido publicado por la plataforma robtica tras una calibración del sistema estéreo que debe disponer. El mensaje es una imagen de disparidad asociada con la imagen de la escena en el instante actual.

El método de subscripción a los mensajes difiere del modo clásico, pues se necesita que los dos mensajes lleguen de manera sincronizada para asegurar que las dos imágenes se corresponden con la misma escena. De este modo la llegada de los dos mensajes sincronizados es el evento que genera el procesado de la rutina de asociación de datos.

Los pasos seguidos para realizar la asociación de marcas son los siguientes:

1. Consulta de la posición del robot.

2. Extracción de marcas.

3. Asociación con el conjunto existente de escenas.

4. Creación de una nueva escena.

5. Publicación de resultados.

Consulta de la posición del robot Es necesario consultar al nodo de visual_iSAM la posición actual del robot, pues las marcas deben ser marcadas como vistas desde dicha posición. Para esto, el nodo visual_iSAM tiene un servidor a tal efecto y el nodo scenes_detector se declara como cliente de dicho servicio.

Extracción de marcas Las tareas asociadas con la extracción de marcas componen la parte principal de procesado de la imagen de la escena. Primero se extraen los puntos característicos de la imagen a través de un detector de marcas. En este caso se usa el detector Good Features to Track de OpenCV, que implementa el método [43]. Se extrae un descriptor de cada una de las marcas detectadas. Del mismo modo que en el caso del detector, OpenCV dispone de varios extractores de descriptores, usándose BRIEF como descriptor para implementar el algoritmo. Se asocia por último un índice de valor negativo a cada una de las marcas para indicar que no ha sido ni asociada con alguna marca del mapa ni añadida al mapa. El conjunto de marcas extraídas con sus descriptores, sus índices y la imagen definen la escena.

Asociación con el conjunto existente de escenas Esta etapa realiza la asociación propiamente dicha. El nodo scenes_detector guarda un vector de escenas que representan vistas clave del recorrido del robot. Cuando este vector contiene ya elementos se compara la escena actual tal como se ha definido con el vector, buscando cual de las escenas almacenadas se asemeja más a la actual. Realizar la comparación comprende dos etapas diferenciadas. La primera es realizar una asociación por mínima distancia haciendo uso de los descriptores de las dos escenas que están siendo comparadas. Acto seguido se aplica RANSAC como método de asociación conjunta para eliminar falsos positivos.

Para definir la mejor escena han de cumplirse dos premisas: Que la cantidad de asociaciones sea mayor de 10 y a su vez que se den la mayor cantidad de asociaciones de entre todas las escenas comparadas.

En vez de recorrer todo el vector para calcular la mejor escena, cosa que según fuera creciendo el vector de escenas haría más lento el procesado de asociación, se ha elegido guardar la escena que se está viendo y recorrer el vector hacia la siguiente escena de la lista, luego a la anterior, y así sucesivamente hasta encontrar la primera escena del vector que de más de 10 asociaciones con la escena actual.

En caso de que ninguna escena del vector consiga asociar 10 o más marcas con la escena actual, se selecciona como mejor escena aquella que haya conseguido la mayor cantidad de asociaciones.

Tanto si se ha conseguido detectar una escena del mapa como escena que describe la vista actual del robot como si no se han conseguido las 10 asociaciones, se asignan los valores de índice de cada una de las marcas asociadas de la escena perteneciente al vector como índice de dicha marca en la escena actual. Se comparan los índices de las asociaciones con la mejor escena y con la última escena asociada del vector. En caso de que haya algún índice que sea diferente en las dos asociaciones entre escenas, se añade el par de índices como índices a fusionar al vector de índices que se enviará como parte del resultado de la asociación de datos.

Por último se recorre las marcas extraidas en la escena actual observando su valor de índice asociado. Aquellas marcas cuyo índice haya cambiado a un valor positivo (los índices se inicializan siempre con un valor negativo) se seleccionan como marcas asociadas al vector de marcas medidas que se enviará como parte del resultado de la asociación de datos.

Creación de una nueva escena En el caso de que la mejor escena no haya conseguido al menos 10 asociaciones, se añade la escena al vector de escenas en la posición siguiente a la anterior escena vista de las ya existentes en el vector. Se busca entonces en la imagen de disparidad el valor asociado de disparidad de cada marca que no esté asociada previamente a ningún punto del mapa y se inicializa con un índice nuevo. Se genera con la información de posición en la imagen y disparidad la marca que se añadirá al vector de marcas que hay que incluir como nuevas al

mapa del SLAM. Este vector de marcas nuevas se enviará como la última parte del resultado de asociación de datos.

Publicación de resultados Los tres vectores mencionados se encapsulan dentro de un mensaje que se publica. Todos aquellos nodos que estén subscritos al mensaje lo recibirán.

Paquete points_publisher

Dentro de este paquete se encuentra el nodo points_publisher, que transforma los mensajes publicados por los nodos del paquete visual_isam para transformar su información en nubes de puntos, de manera que tanto los recorridos del robot como las posiciones de las marcas puedan representarse en el simulador RVIZ.

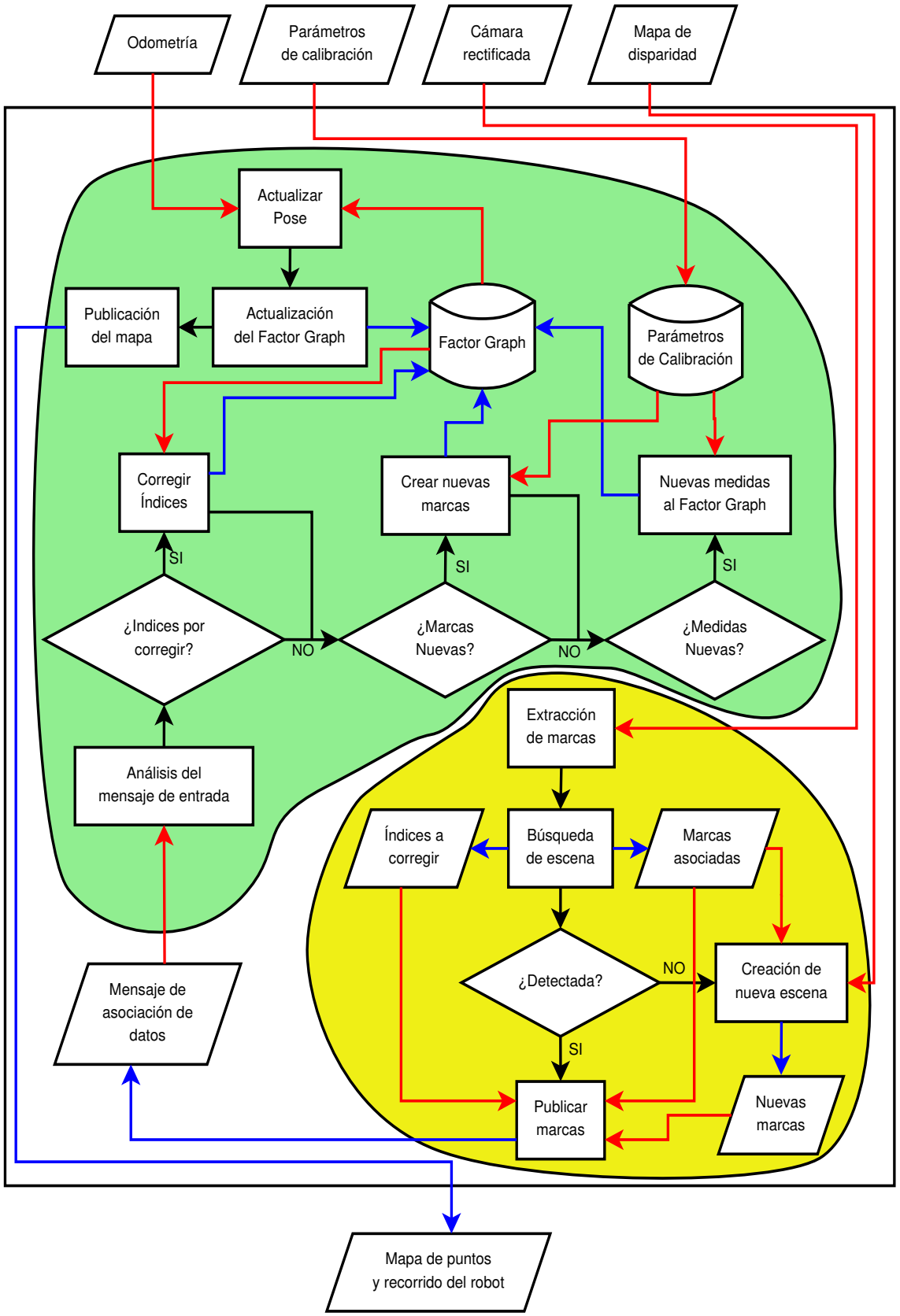

Figura 4.13: Diagrama funcional de la propuesta para la solución del SLAM

Capítulo 5

Resultados

5.1. Introducción

Este trabajo trata de resolver el problema del SLAM mediante visión artificial en entornos interiores. Dado que a fecha de publicación de este proyecto no se dispone de un ground truth en interiores lo suficientemente preciso, se propone utilizar un análisis visual para comparar los algoritmos EKF SLAM e iSAM. Por ello, se comparará la estimación de la posición dada por la odometría con respecto a la dada por los filtros. Para realizar esta comparación se dibujará en el entorno gráfico RVIZ las trayectorias seguidas por el robot con y sin el algoritmo. Al no tener la posibilidad de medir la posición real del robot en cada momento se intentará en cada experimento que la posición inicial y final del robot sea la misma, de manera que se pueda estimar con un bajo margen de equivocación el error acumulado por el robot con las dos aproximaciones. Se realizarán las pruebas en dos entornos diferentes, haciendo uso de una grabación proporcionada por ROS del recorrido realizado por el robot en cada uno de estos entornos, de manera que los mensajes de entrada a ambos programas sean los mismos.

Problemas de sincronismo

Se ha encontrado un problema particular a la hora de asociar la información del movimiento del robot y la proveniente de las cámaras, así como asociar las imágenes de las dos cámaras entre si. Este problema radica en el hecho que el driver de las cámaras USB no proporciona las imágenes del instante actual sino de un tiempo ya pasado, aparte de no llevar una sincronía en el tiempo entre las imágenes de las dos cámaras. Para solucionar el fallo que concierne a la sincronía de las cámaras se ha realizado un nodo en ROS que permite sincronizar las imágenes con bastante exactitud. Aun así giros bruscos del robot sobre si mismo afectan negativamente a la inclusión de nuevas marcas al mapa debido a este inconveniente. Para solucionar el problema de falta de sincronía en el tiempo entre las imágenes y la odometría no se ha optado por ninguna solución particular y se ha dejado que el propio estimador bayesiano corrija los estados del sistema a pesar de estas variaciones.

5.2. Entorno 1: Laboratorio

El laboratorio es un escenario de 7m de longitud y otros 7m de anchura con luz ambiente que entra por una ventana. Existen mesas y cajones, así como sillas y ordenadores por alrededor. Una persona graba al robot mientras realiza su recorrido. El robot la detecta y dicha persona se mueve, pudiendo provocar errores por asociar puntos móviles. El robot realiza una trayectoria ovalada alrededor de una mesa que hay justo en el centro del laboratorio.

5.2.1. Análisis de la asociación de datos

Este análisis estudia el comportamiento del método propuesto para realizar la tarea de asociación de datos, tanto en la cantidad de información extraida del entorno como en el tiempo de cómputo empleado para realizar su función.

Las figuras 5.1, 5.2 y 5.3 corresponden a las asociaciones entre la imagen vista por el robot en tres momentos diferentes y la mejor escena seleccionada del vector de escenas características almacenadas.

En estas imágenes se observa, gracias a las líneas que unen las asociaciones entre la imagen del robot y la del vector de escenas, que para asociar las marcas entre escenas solo hace falta

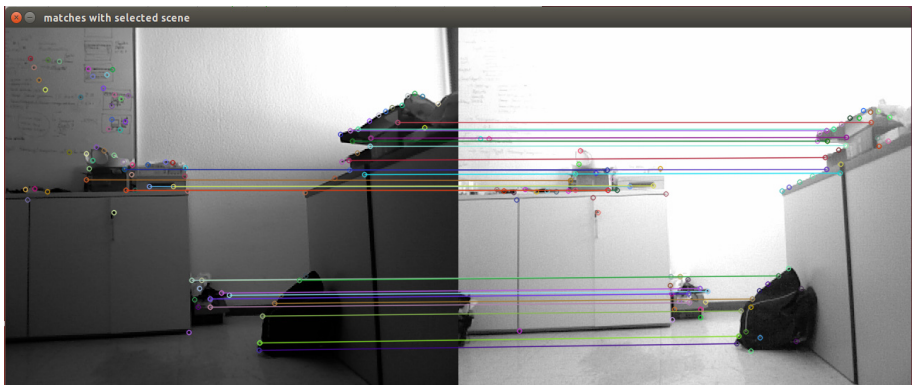

Figura 5.1: Vista de la cámara y asociación de puntos. Posición 1 dentro del laboratorio

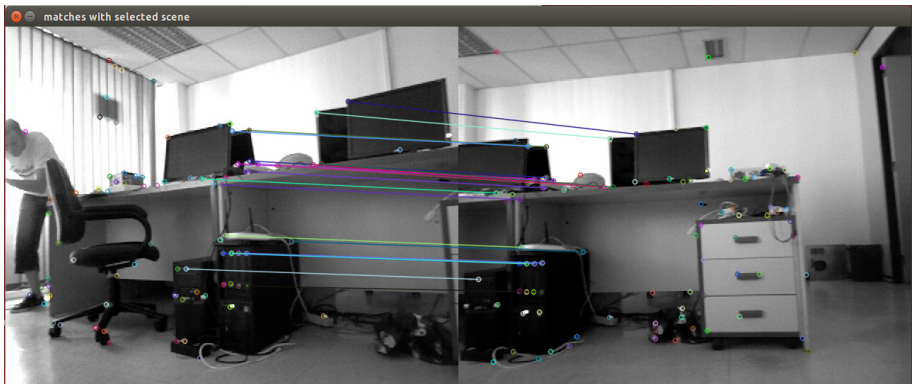

Figura 5.2: Vista de la cámara y asociación de puntos. Posición 2 dentro del laboratorio

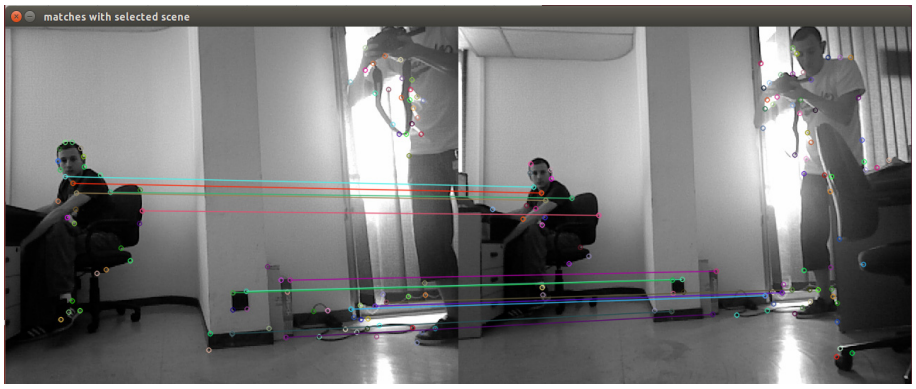

Figura 5.3: Vista de la cámara y asociación de puntos. Posición 3 dentro del laboratorio

encontrar una transformación afín que lleve los puntos de una imagen a otra. Esta es la transformación que se consigue con el algoritmo RANSAC, mediante la cual aquellas marcas que no tienen línea de unión han sido descartadas por no ajustarse a dicha transformación. De estas imágenes también se puede extraer información sobre la cantidad de marcas que se extraen de una escena, así como las marcas pertenecientes al mapa que son detectadas en la escena vista por el robot.

La cantidad de marcas que se extraen de la escena que observa el robot en cada momento se representa en la figura 5.4.

Con los resultados obtenidos se infiere que el algoritmo es capaz de detectar un número

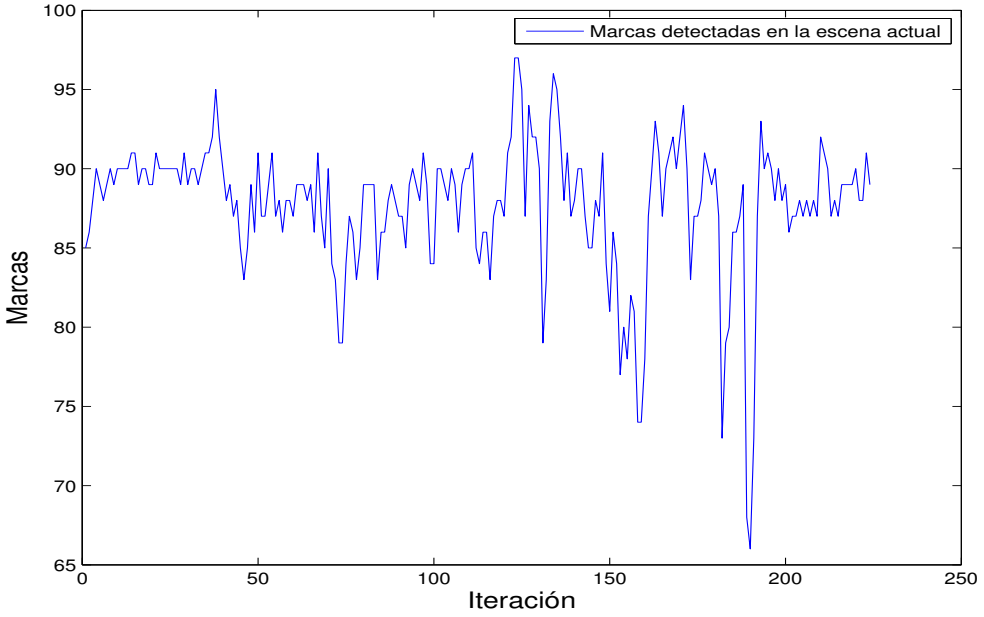

Figura 5.4: Marcas detectadas en la imagen vista por el robot. Laboratorio

suficientemente amplio de marcas como para representar el entorno, lo que llevará a un número de escenas características del mapa más pequeño, ya que éstas serán más fácilmente detectables debido a ese alto número de marcas.

La cantidad de marcas que consiguen asociarse con las marcas almacenadas en el mapa se representa en la figura 5.5

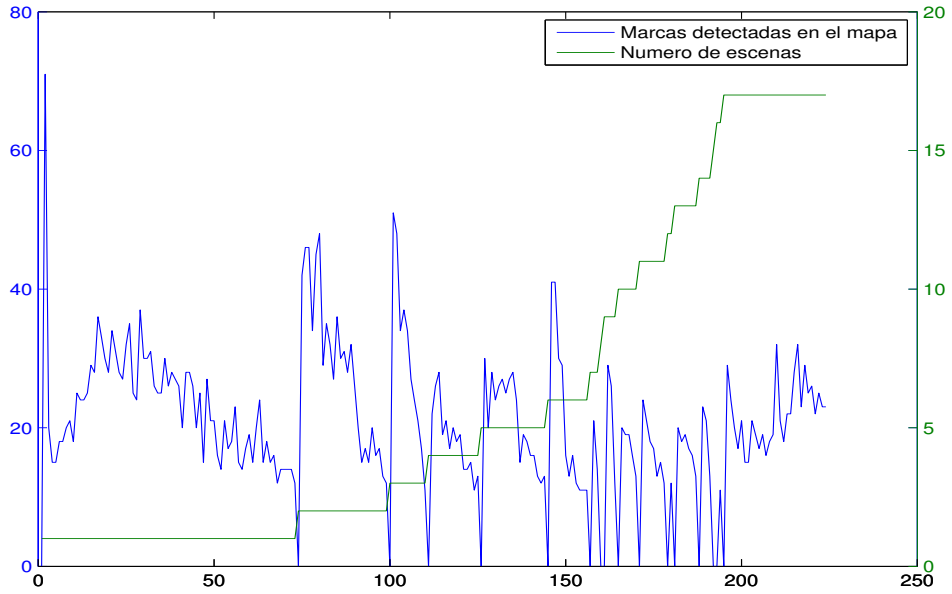

Figura 5.5: Asociación con el mapa de puntos. Laboratorio

La asociación de puntos entre la escena vista y el mapa ronda las 20 marcas. Es con estas asociaciones con las que el SLAM corregirá su posición. Cuando la asociación de puntos baja del umbral de 10, se introduce una nueva escena al vector de escenas características, que a su vez añade más puntos nuevos al mapa.

El tiempo de cómputo del algoritmo de asociación de datos según va avanzando la exploración de la zona se observa en la figura 5.6.

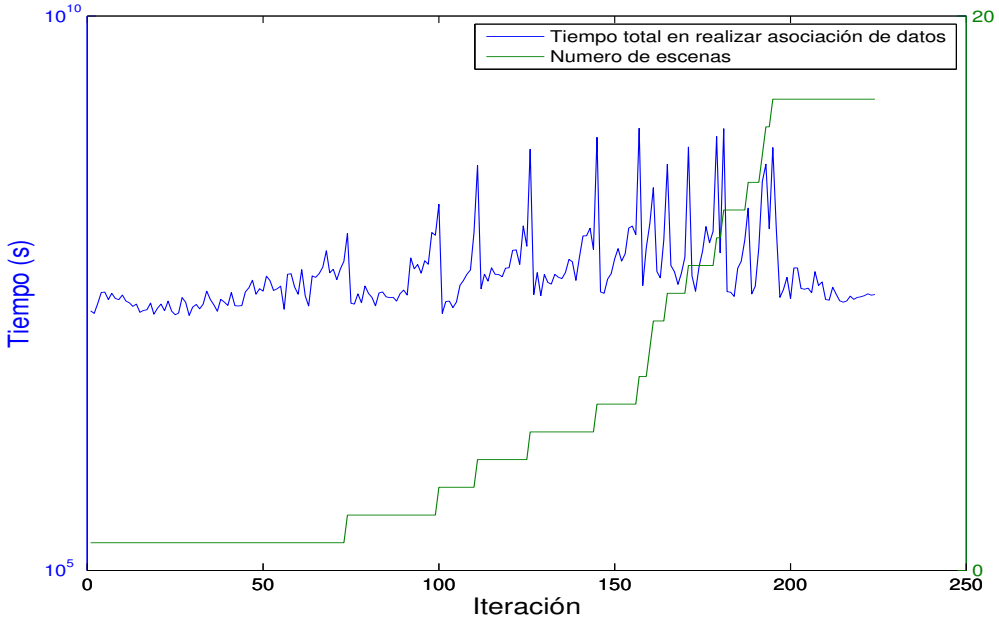

Figura 5.6: Tiempo de cómputo en realizar la asociación de datos. Laboratorio

Típicamente este tiempo está por debajo de una décima de segundo, salvo cuando se añade una escena al vector de escenas, que dispara ese tiempo por encima de 1 segundo cuando la cantidad de escenas en el vector supera las 15.

En el caso particular del iSAM, al realizarse la asociación de datos en un proceso paralelo a la actualización del Factor Graph y tener guardada la posición del robot en el momento de la captura de las imágenes, este retardo no es muy grave, ya que una vez pueda ser incluido en el Factor Graph, su etapa de actualización corregirá todas las posiciones que han ido llegando debido a la odometría desde la posición de referencia de las imágenes y la posición del momento en el que se realiza la actualización. Aun así, este retardo descartará medidas del entorno mientras se esté haciendo el cómputo de las asociaciones, dejando zonas del entorno sin corrección, con lo que interesa mantenerlo lo más bajo posible.

Para el caso del EKF SLAM se puede seguir realizando en un proceso paralelo la asociación de datos, pero el proceso encargado de la gestión del mapa deberá esperar a disponer del mensaje de asiciación de datos para realizar la actualización, teniendo que unir el desplazamiento que haya sufrido el robot en ese intervalo de tiempo en un único mensaje de odometría, separándose más el modelo real de movimiento del robot de su versión linealizada y por tanto haciendo más probable la divergencia del estimador.

5.2.2. Análisis de la gestión del mapa y posición del robot

En esta sección se presenta el análisis visual de los algoritmos de SLAM así como de sus tiempos de cómputo.

En la figura 5.7 se muestran tanto los resultados obtenidos con la odometría como con el iSAM. La ruta coloreada en verde es la ruta estimada haciendo uso únicamente de la odometría. La ruta coloreada en rojo es la ruta corregida por el algoritmo.

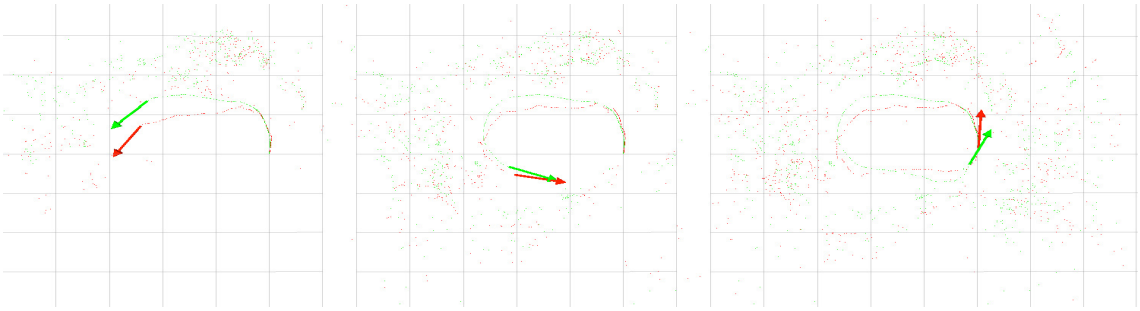

Figura 5.7: Laboratorio. Comparación trayectoria Odometría iSAM

En la figura 5.8 se puede ver el resultado obtenido con el EKF. La ruta coloreada en verde es la ruta estimada haciendo uso únicamente de la odometría. La ruta coloreada en rojo es la ruta corregida por el algoritmo.

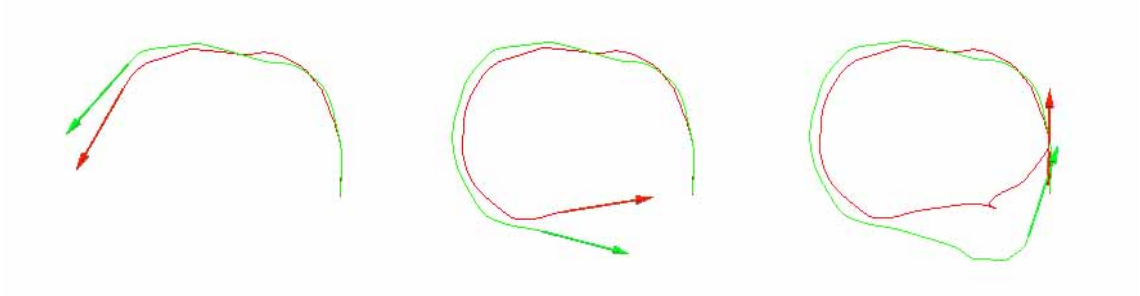

Figura 5.8: Laboratorio. Comparación trayectoria Odometría EKF

Ambos recorridos son semejantes, llegando a la misma posición final y corrigiendo el error cometido por la odometría. Esto quiere decir que, en cuanto a la creación del mapa y localización del robot los dos enfoques son válidos. En el caso del uso de *Smoothing and Mapping* los errores que pueda cometer el propio SLAM en la corrección de la posición se suavizan al optimizarse la trayectoria completa del robot, no solo la posición final como ocurre con un enfoque basado en filtros, como es el caso del EKF.

El tiempo de cómputo por parte del *Smoothing and Mapping* en añadir nuevas mediciones de marcas al Factor Graph se representa en la figura 5.9 con su eje de coordenadas y en escala logarítmica.

Este tiempo está siempre por debajo de $1ms$. Está bien que el tiempo que se gasta en este procesado sea bajo y esté acotado, ya que donde se va a necesitar realizar más cómputos es en la actualización del Factor Graph. Esta etapa podría ser equivalente a la etapa de predicción del EKF, cuyo tiempo de cómputo también resultó ser del mismo orden que el caso expuesto.

El tiempo de cómputo por parte del *Smoothing and Mapping* en realizar la tarea de actualización del Factor Graph se representa en la figura 5.10 con su eje de coordenadas y en escala

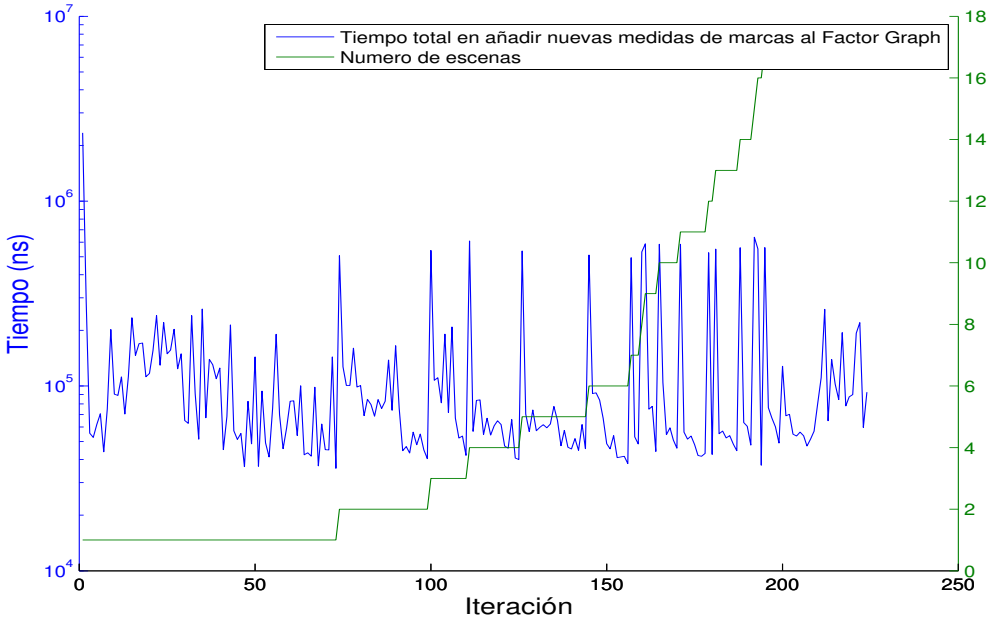

Figura 5.9: Tiempo de cómputo en añadir nuevas medidas al Factor Graph. Laboratorio

logarítmica. Este tiempo se compara en la misma gráfica con su equivalente en el EKF, que se corresponde a la etapa de actualización del filtro y adición de nuevas marcas al mapa.

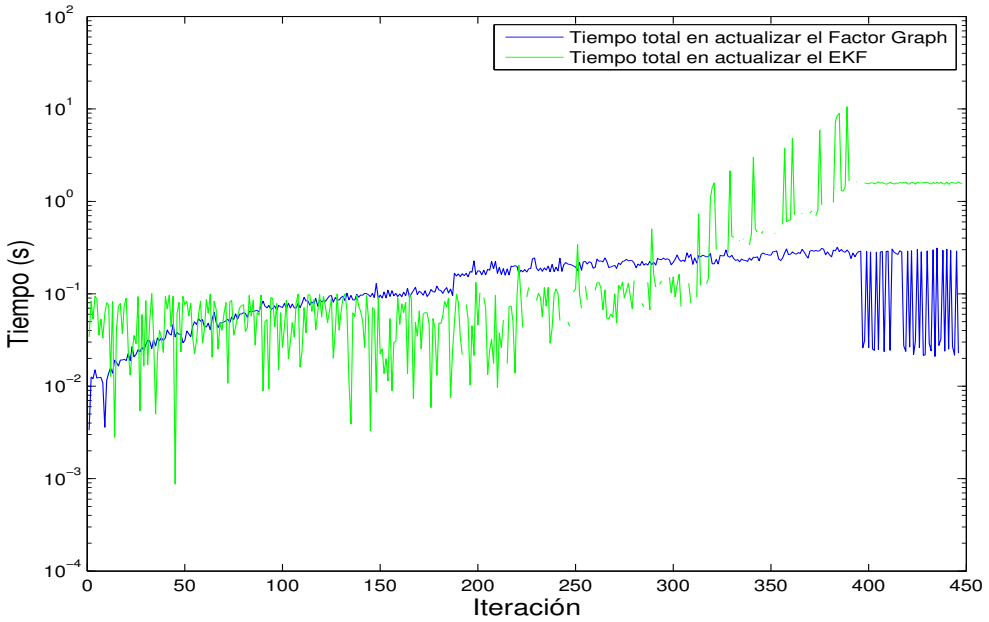

Figura 5.10: Tiempo de cómputo en realizar la etapa de actualización. Laboratorio

En el caso de iSAM, el tiempo crece de manera lineal con la cantidad de marcas y posiciones que contiene el mapa, asemejándose a la gráfica vista en la figura 2.12. , sin embargo, para el caso del EKF la evolución del tiempo no depende de las posiciones del robot, sino solo de las marcas, pero su tiempo de cómputo es de $O(n^3)$, derivando en que cuanto más crezca el mapa

mucho peor será su respuesta.

Por ello el método basado en *Smoothing and Mapping* es más rápido que el método basado en el EKF realizando la etapa de actualización cuando se llega a sus etapas finales. No solo es más rápido, sino que es capaz de gestionar un mayor número de marcas. En este caso las marcas finales de cada mapa son 1340 en el caso del *Smoothing and Mapping* y 333 en el caso del EKF. Este aumento en las marcas que se han añadido al mapa se puede extrapolar a que, si se es restrictivo en la adición de marcas al mapa se podría gestionar un entorno al menos cinco veces mayor, o bien que ante un entorno del mismo tamaño, como es el caso, el describirlo con más marcas permite una localización precisa a pesar de ligeras modificaciones como puede ser objetos en movimiento o la aparición de objetos como sillas o mesas.

Los picos que se observan en el tiempo de cómputo del EKF se corresponden con las etapas de adición de nuevas marcas. La complejidad en realizar este procesado también es $O(n^3)$, pero se realiza de manera iterativa sobre un gran número de marcas.

5.3. Entorno 2: Sótano

El sótano es una habitación cerrada, sin luz natural, de 7 metros de largo por 5 de ancho. Se encuentran en los alrededores cuadros, productos de limpieza y alguna silla plegable. el robot realiza una trayectoria ovalada bordeando la habitación. Esta trayectoria se realiza dos veces.

5.3.1. Análisis de la asociación de datos

Las figuras 5.11, 5.12 y 5.13 corresponden a las asociaciones entre la imagen vista por el robot en ese momento y la mejor escena seleccionada del vector de escenas características almacenadas.

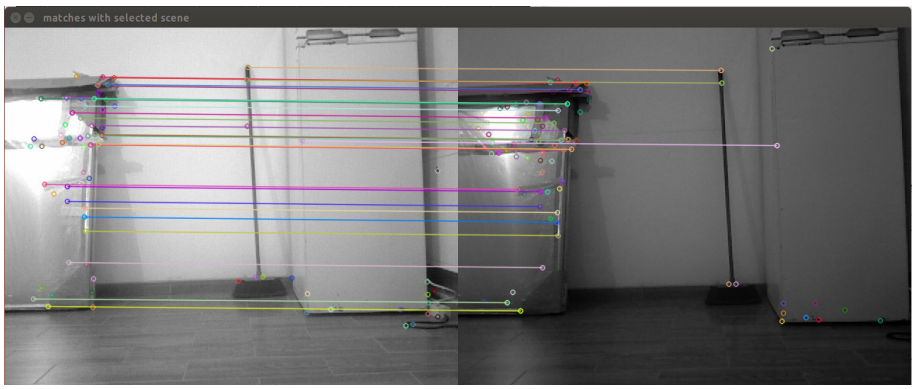

Figura 5.11: Vista de la cámara y asociación de puntos. Posición 1 en el sótano

Tal como ocurre en el primer escenario, se realiza una asociación correcta entre escenas, sin falsos positivos que pudieran llevar a la divergencia del resultado. Se puede apreciar también que el algoritmo es robusto ante cambios de luminosidad y de rotación en las imágenes.

La cantidad de marcas que se extraen de la escena que observa el robot en cada momento se representa en la figura 5.14.

En este escenario se extraen de media las mismas marcas naturales. La diferencia radica en que en el primer escenario la cantidad de marcas extraída se mantiene más o menos constante

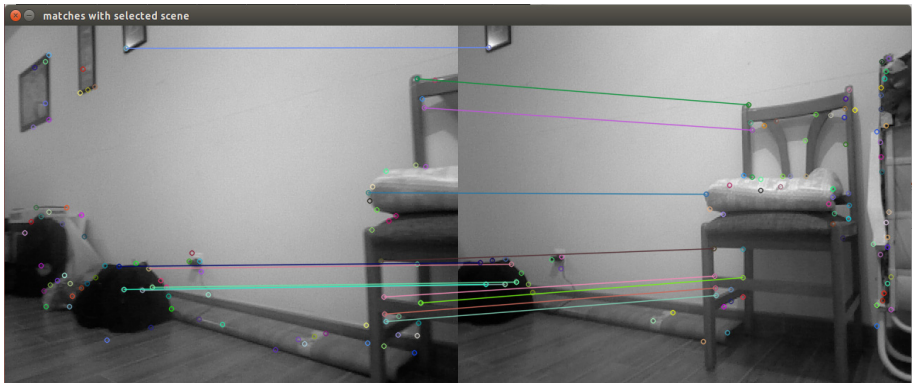

Figura 5.12: Vista de la cámara y asociación de puntos. Posición 2 en el sótano

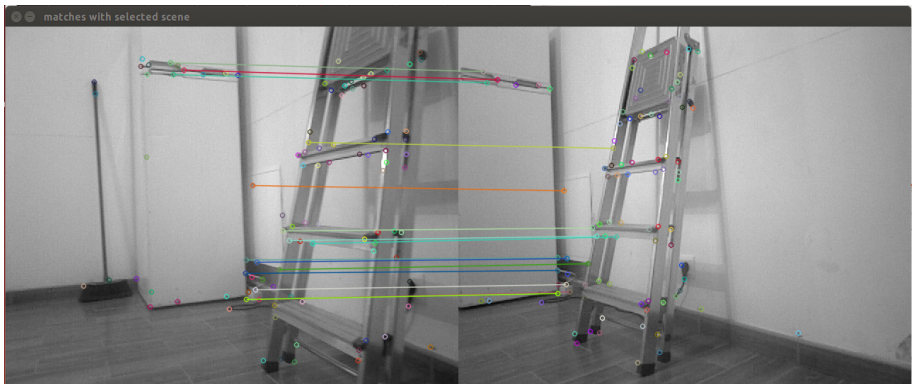

Figura 5.13: Vista de la cámara y asociación de puntos. Posición 3 en el sótano

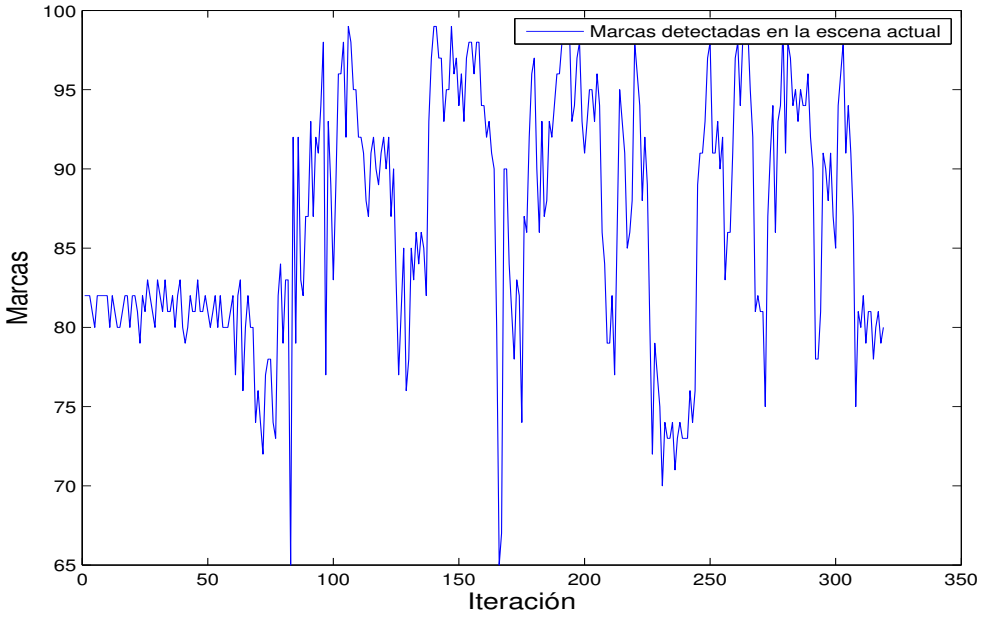

Figura 5.14: Marcas detectadas en la imagen vista por el robot. Sótano

por todo el recorrido mientras que en este escenario las varianza en cuanto a las marcas extraídas

es mayor. Aun a pesar de pasar por zonas con menos información, esta es suficiente para que se pueda realizar la asociación como se verá más adelante.

La cantidad de marcas que consiguen asociarse con las marcas almacenadas en el mapa se representa en la figura 5.15

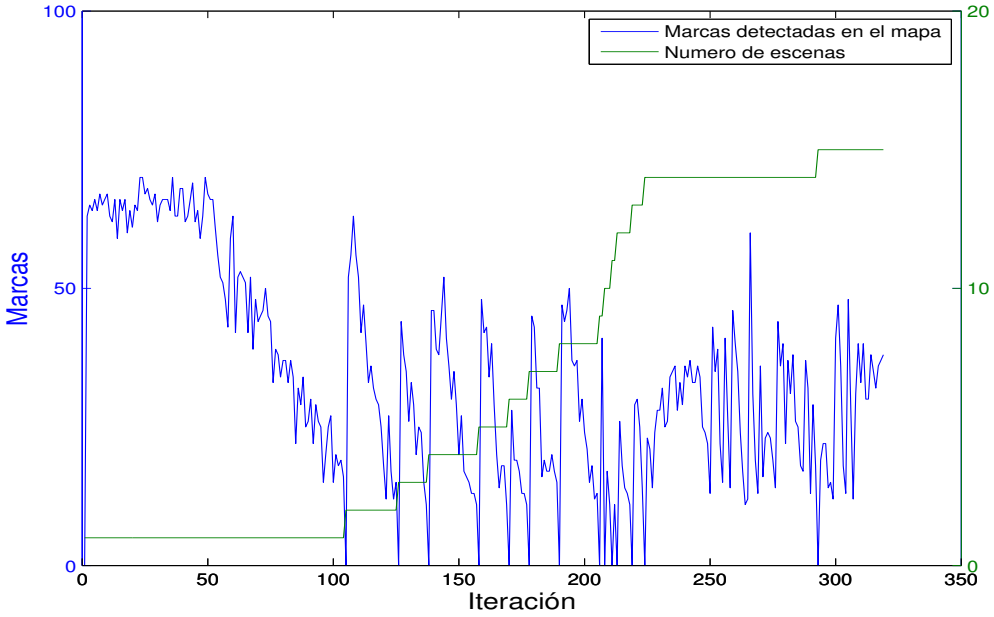

Figura 5.15: Asociación con el mapa de puntos. Sótano

La asociación de puntos entre la escena vista y el mapa ronda las 30 marcas de media, que es un valor aun más alto que en el primer recorrido.

El tiempo de cómputo del algoritmo de asociacin de datos según va avanzando la exploración de la zona se observa en la figura 5.16.

Igual que en el primer escenario, el tiempo está acotado por debajo de un segundo salvo cuando se necesita añadir una escena al vector de escenas, momento en el que poco a poco va aumentando cada vez más. Este comportamiento es el que limitará el tamaño y características del entorno sobre el que se quiere mover el robot.

5.3.2. Análisis de la gestión del mapa y posición del robot

En la figura 5.17 se muestran tanto los resultados obtenidos con la odometría como con el iSAM. En esta prueba se dan dos vueltas al entorno.

El motivo de dar dos vueltas es para que se pueda apreciar que, una vez se recorre una zona de la que ya se disponen marcas en el mapa, el error en la posición del robot no crece y las marcas almacenadas se siguen asociando correctamente, esto es, con el fin de comprobar el funcionamiento de los algoritmos ante el cierre de lazo.

En la figura 5.18 se puede ver el resultado obtenido con el EKF. La ruta coloreada en verde es la ruta estimada haciendo uso únicamente de la odometría. La ruta coloreada en rojo es la ruta corregida por el algoritmo.

Se observa que el comportamiento en este escenario es similar al primer escenario probado.

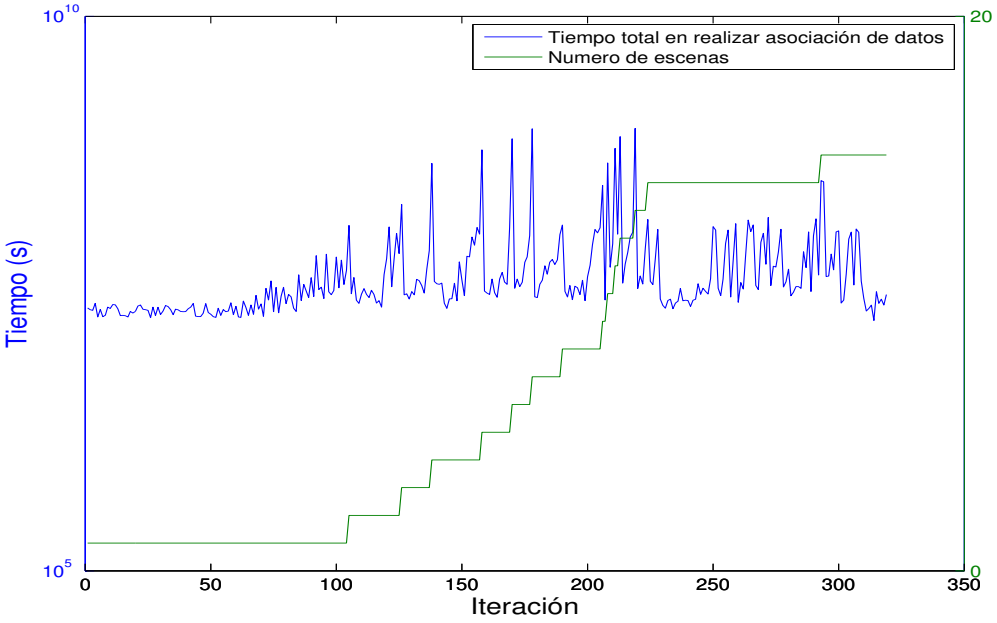

Figura 5.16: Tiempo de cómputo en realizar la asociación de datos. Sótano

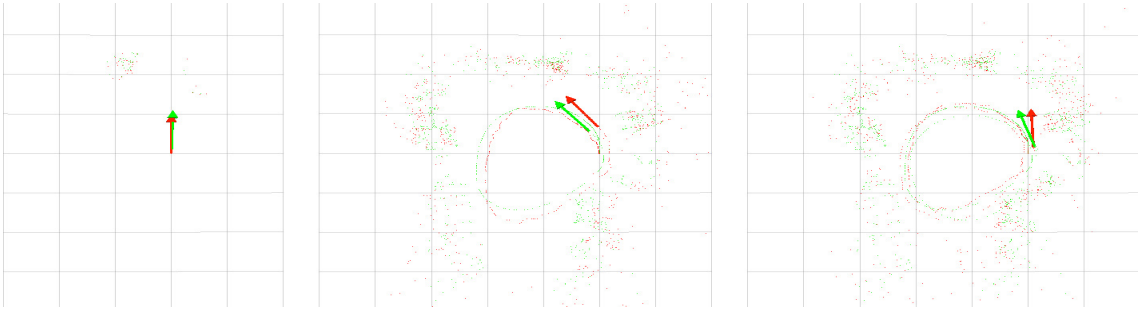

Figura 5.17: Sótano. Comparación trayectoria Odometría iSAM

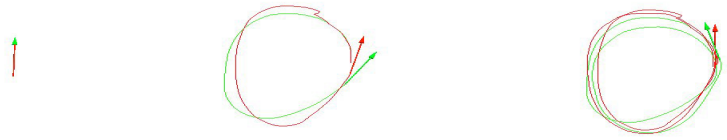

Figura 5.18: Sótano. Comparación trayectoria Odometría EKF

El tiempo de cómputo por parte del *Smoothing and Mapping* en añadir nuevas mediciones de marcas al Factor Graph se representa en la figura 5.19 con su eje de coordenadas y en escala logarítmica.

Este tiempo está siempre por debajo de $1ms$, obteniendo resultados similares al escenario 1.

El tiempo de cómputo por parte del *Smoothing and Mapping* en realizar la tarea de actua-

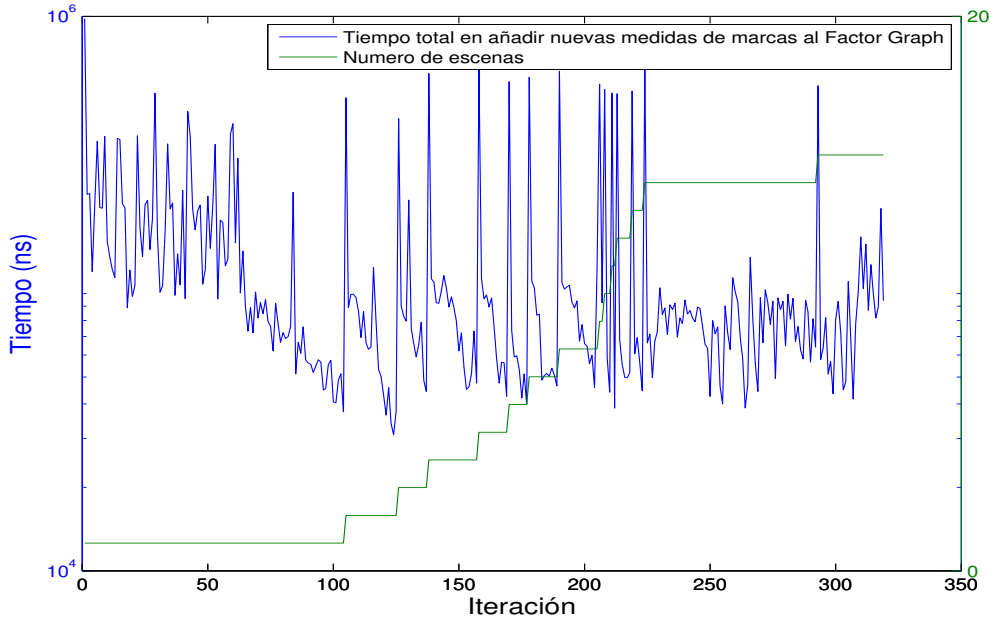

Figura 5.19: Tiempo de cómputo en añadir nuevas medidas al Factor Graph. Sótano

lización del Factor Graph se representa en la figura 5.20 con su eje de coordenadas y en escala logarítmica. Este tiempo se compara en la misma gráfica con su equivalente en el EKF, que se corresponde a la etapa de actualización del filtro y adición de nuevas marcas al mapa.

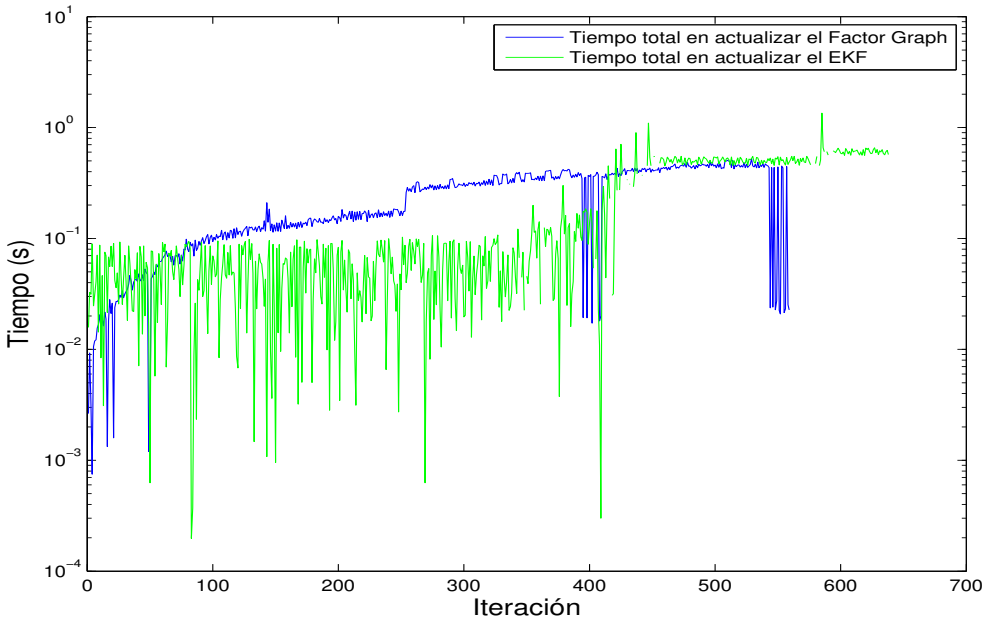

Figura 5.20: Tiempo de cómputo en realizar la actualización del Factor Graph. Sótano

La evolución del tiempo en ambos casos es similar al primer escenario. Sin embargo se puede observar que los resultados de tiempo empleado son similares entre el iSAM y el EKF salvo por un detalle, que es que las marcas incluidas en el mapa para el caso de *Smoothing and Mapping* es

de 1196 mientras que en el caso del EKF el mapa dispone de 143 marcas. Esto quiere decir que con el mismo tiempo de procesado el algoritmo basado en *Smoothing and Mapping* es capaz de manejar un mapa aproximadamente 10 veces mayor que en el caso del EKF. Los picos presentes en la evolución del tiempo del EKF son debidos a la adición de nuevas marcas y escenas al mapa, tal como se explicó con el primer escenario.

5.4. Análisis numérico de los resultados

En esta sección se realizará una comparativa numérica entre el EKF SLAM y el iSAM. Se compararán las diferencias de ajustes que se han realizado sobre la etapa de asociación de datos para acabar con los resultados de tiempo de ejecución de ambas propuestas.

Las tablas 5.1 y 5.2 resumen los datos obtenidos del sistema de visión estereoscópica en los dos experimentos, usando los ajustes particulares para el EKF SLAM y para el *Smoothing and Mapping* respectivamente.

Experimento	**Cantidad total de imágenes**	**Cantidad de imágenes almacenadas**	**Cantidad de puntos del mapa**	**Puntos asociados del mapa**
Laboratorio	756	27	333	20
Sótano	1026	18	143	17

Tabla 5.1: Tabla de resultados de las imágenes para el EKF

Experimento	**Cantidad total de imágenes**	**Cantidad de imágenes almacenadas**	**Cantidad de puntos del mapa**	**Puntos asociados del mapa**
Laboratorio	756	17	1340	21
Sótano	1026	15	1196	35

Tabla 5.2: Tabla de resultados de las imágenes para el iSAM

En el caso del EKF hubo que ser muy estricto a la hora de decidir incluir un punto al mapa, ya que el límite de puntos en el que el mapa se podía gestionar con agilidad era de aproximadamente 200. En el caso del iSAM se han llegado a gestionar hasta 1300 marcas sin que el tiempo de procesado llegara a ser un inconveniente.

La diferencia en la cantidad de escenas clave que representan la trayectoria viene de la adaptación y optimización del proceso de asociación de datos a las características diferentes del propio estimador bayesiano.

En las tablas 5.3 y 5.4 se representan los tiempos medios de cómputo obtenidos con el uso del filtro de Kalman extendido y con el *Smoothing and Mapping* respectivamente en los tramos indicados en cada tabla.

Si se compara el tramo de marcas menor a 300, que es compartido en los dos análisis, se extrae que la etapa de asociación de datos se comporta igual, ya que la nueva versión comparte la idea general de funcionamiento que la implementada en el proyecto de origen. Sin embargo, la etapa de actualización del mapa sí presenta diferencias sustanciales. La actualización del Factor Graph, que equivale a la corrección del EKF, es 22 veces más rápida que la etapa de corrección. Para el caso de añadir nuevas marcas y medidas al mapa, los tiempos en este tramo son similares, sin embargo, si se observa la evolución de ambos tiempos, en el caso del *Smoothing*

Etapa	**<100 marcas**	**<200 marcas**	**<300 marcas**	**>300 marcas**
Asociación de marcas	0,11s	0,14s	0,22s	0,25s
Corrección EKF	0,27s	0,57s	0,81s	1,56s
Adición de nuevas marcas	0,24s	1,05s	10,09s	37,12s

Tabla 5.3: Resumen de tiempos dedicados a cada paso de la etapa de corrección del estado del sistema. EKF

Etapa	**<300 marcas**	**<700 marcas**	**<1000 marcas**	**>1000 marcas**
Asociación de marcas	22ms	0,24s	1,1s	2,0s
Actualización del Factor Graph	0,07s	0,16s	0,21s	0,31s
Adición de nuevas medidas	0,25ms	0,25ms	0,25ms	0,25ms

Tabla 5.4: Resumen de tiempos dedicados a cada paso de la etapa de corrección del estado del sistema. Smoothing and Mapping

and Mapping el tiempo de cómputo se mantiene constante, mientras que para el EKF aumenta exponencialmente.

Añadir nuevas marcas al mapa, en el caso del uso del filtro de Kalman extendido era el mayor de los problemas con que se encontraba dicha solución al SLAM. En el caso de la solución propuesta en este trabajo, donde se realiza la tarea más ardua de computación es a la hora de actualizar el Factor Graph. Este tiempo se mantiene por debajo de un segundo durante los dos experimentos. Sin embargo, la etapa de asociación de datos en la adaptación al iSAM da la impresión de ofrecer peores resultados, pero hay que tener en cuenta el aumento en la cantidad de marcas del mapa que se asocian. Este aumento de marcas en el mapa permite una mayor robustez a la hora de eliminar falsos positivos en la asociación de datos, que llevarían a una divergencia del resultado del algoritmo con respecto a la posición real del robot.

Capítulo 6

Conclusiones

6.1. Conclusiones y Trabajos Futuros

Tal como se ha podido observar en los resultados obtenidos se ha conseguido el objetivo del trabajo, que era mejorar el comportamiento con respecto al tiempo y las limitaciones de tamaño de mapa que se encontraron en el enfoque de EKF SLAM. Se puede concluir por tanto que el actualizar el SLAM haciendo uso de *Smoothing and Mapping*, tal como se está haciendo con muchos de los enfoques actuales para solucionar la localización y mapeo simultáneos ha dado como resultado una aplicación que permite realizar la tarea prevista en entornos interiores con una cantidad de puntos al menos cinco veces mayor que en el caso del EKF.

Con este enfoque el cuello de botella ha pasado a ser el método de asociación de datos. El inconveniente viene específicamente por cada vez que se observe una escena diferente, que requiere ejecutar el algoritmo RANSAC en varias ocasiones. Para mejorar el tiempo de procesado se abren varias opciones: Minimizar la cantidad de veces que hay que ejecutar RANSAC a partir de una ordenación más inteligente del vector de escenas, hacer uso de GPU's o implementar la opción de multitarea para aprovechar la capacidad de los microprocesadores actuales con su alta cantidad de núcleos y realizar de manera simultánea varias asociaciones entre la escena actual y un conjunto de escenas del vector.

El uso de visión artificial y odometría para realizar el SLAM genera un mapa con bastante incertidumbre debido a los errores inherentes a cada uno de los sensores usados, sobretodo en el caso de la plataforma usada que dispone de un sistema sensorial de bajo coste. Añadir otros sensores que proporcionen distinta información sobre el entorno conllevaría a un mejor resultado en la creación del mapa a la par que disponer de un mapa más rico en cuanto la información que se extraería de la zona. Por ejemplo, las marcas presentes en un entorno interior como es el caso en el que se ha probado el algoritmo generan como mapa una nube de puntos que es dificil de interpretar como mapa por parte de un ajente externo humano. Si se añadiera un sensor laser de barrido se podría generar el mapa no solo como una nube de puntos sino con una descripción más clara de los posibles obstáculos y paredes que se hallan en el entorno. Se puede partir desde el punto en el que se queda este trabajo para tratar de fusionar distintos tipos de sensores, de manera que se consigan generar mapas más precisos y que definan mejor el entorno en el que se encuentra el robot.

Capítulo 7

Bibliografía

Bibliografía

[1] D. Julian, *Construcción de Mapas Probabilísticos mediante sensores de visión y laser a bordo de un robot.* Universidad de Alcala, 2012.

[2] M. Kaess, A. Ranganathan, and F. Dellaert, "iSAM: Incremental smoothing and mapping," *IEEE Transactions on Robotics*, vol. 24, pp. 1365–1378, 2008.

[3] R. Smith, M. Self, and P. Cheeseman, "Estimating uncertain spatial relationships in robotics," in *Proc. IEEE Int. Conf. Robotics and Automation*, vol. 4, 1987.

[4] H. F. Durrant-Whyte, "Uncertain geometry in robotics," *IEEE J. Robot. Automat.*, vol. 4, no. 1, pp. 23–31, 1988.

[5] N. Ayache and O. Faugeras, "Building, registrating, and fusing noisy visual maps," *International Journal Of Robotics Research*, vol. 7, no. 6, pp. 45–65, 1988. [Online]. Available: http://ijr.sagepub.com/cgi/doi/10.1177/027836498800700605

[6] J. L. Crowley, "World modeling and position estimation for a mobile robot using ultrasonic ranging," in *Proc. Conf. IEEE Int Robotics and Automation*, 1989, pp. 674–680.

[7] R. Chatila and J. Laumond, "Position referencing and consistent world modeling for mobile robots," in *Proc. IEEE Int. Conf. Robotics and Automation*, vol. 2, 1985, pp. 138–145.

[8] R. C. Smith and P. Cheeseman, "On the representation and estimation of spatial uncertainty," *The International Journal of Robotics Research*, vol. 5, no. 4, pp. 56–68, 1987. [Online]. Available: http://ijr.sagepub.com/cgi/doi/10.1177/027836498600500404

[9] J. J. Leonard and H. F. Durrant-Whyte, "Simultaneous map building and localization for an autonomous mobile robot," in *Proc. IEEE/RSJ Int. Workshop Intelligent Robots and Systems '91. 'Intelligence for Mechanical Systems IROS '91*, 1991, pp. 1442–1447.

[10] ——, *Directed Sonar Sensing for Mobile Robot Navigation.* Kluwer Academic Publishers, 1992, no. 1. [Online]. Available: http://books.google.com/books?hl=en&lr=&id=r-7q_GWz3FAC&oi=fnd&pg=PR11&dq=Directed+Sonar+Sensing+for+Mobile+Robot+Navigation&ots=Pu7-J_3E01&sig=chM4kjaroCH16dsw6u-eCCsZDwM

[11] W. D. Rencken, "Concurrent localisation and map building for mobile robots using ultrasonic sensors," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems '93 IROS '93*, vol. 3, 1993, pp. 2192–2197.

[12] H. F. Durrant-Whyte, D. Rye, and E. Nebot, "Localisation of automatic guided vehicles," *Proceedings of the 7th International Symposium on Robotics Research*, vol. 25, no. 12, pp. 613–625, 1996. [Online]. Available: http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Localisation+of+automatic+guided+vehicles#0

[13] M. Csorba, "Simultaneous localisation and map building," Ph.D. dissertation, Ieee, 1997. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1013646

[14] M. Csorba and H. F. Durrant-Whyte, *A New Approach to Simultaneous Localisation and Map Building*, 1996.

[15] J. J. Leonard and H. J. S. Feder, "A computationally efficient method for large-scale concurrent mapping and localization," *Simulation*, vol. 9, no. 3, pp. 169–176, 2000. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.36.9767&rep=rep1&type=pdf

[16] J. A. Castellanos, J. M. MartÃnez, J. Neira, and J. D. TardÃ^3*s*, *Experiments in Multisensor Mobile Robot Localization and Map Building*, 1998.

[17] J. A. Castellanos, J. D. Tardos, and G. Schmidt, "Building a global map of the environment of a mobile robot: the importance of correlations," in *Proc. Conf. IEEE Int Robotics and Automation*, vol. 2, 1997, pp. 1053–1059.

[18] J. Guivant, E. Nebot, and S. Baiker, "Localization and map building using laser range sensors in outdoor applications," *Journal of Robotic Systems*, vol. 17, no. 10, pp. 565–583, 2000. [Online]. Available: http://doi.wiley.com/10.1002/1097-4563%28200010%2917%3A10%3C565%3A%3AAID-ROB4%3E3.0.CO%3B2-6

[19] S. B. Williams, P. Newman, G. Dissanayake, and H. Durrant-Whyte, "Autonomous underwater simultaneous localisation and map building," in *Proc. IEEE Int. Conf. Robotics and Automation ICRA '00*, vol. 2, 2000, pp. 1793–1798.

[20] K. S. Chong and L. Kleeman, "Feature-based mapping in real, large-scale environments using an ultrasonic array," *International Journal Of Robotics Research*, vol. 18, no. 1, pp. 3–19, 1999. [Online]. Available: http://www.ingentaselect.com/rpsv/cgi-bin/cgi?ini=xref&body=linker&reqdoi=10.1177/02783649922066033

[21] M. Deans and M. Hebert, *Experimental comparison of techniques for localization and mapping using a bearings only sensor.*, 2000.

[22] J. Hollerbach and D. K. (Eds), *Robotics Research: the Ninth International Symposium*, L. Springer-Verlag, Ed., 2000.

[23] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. J. Leonard, and F. Dellaert, "iSAM2: Incremental smoothing and mapping using the Bayes tree," pp. 216–235, 2012.

[24] P. Newman, "On the structure and solution of the simultaneous localisation and map building problem," Ph.D. dissertation, 1999. [Online]. Available: http://www.frc.ri.cmu.edu/~ssingh/pubs/others/pmn.pdf

[25] G. Dissanayake, H. Durrant-Whyte, and T. Bailey, "A computationally efficient solution to the simultaneous localisation and map building (slam) problem," *Proceedings 2000 ICRA Millennium Conference IEEE International Conference on Robotics and Automation Symposia Proceedings Cat No00CH37065*, no. April 2000, pp. 1009–1014, 2000. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=844732

[26] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, "Fastslam 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges," *International Joint Conference on Artificial Intelligence*, vol. 18, no. 1, pp.

1151–1156, 2003. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.64.3773&rep=rep1&type=pdf

[27] S. Thrun, W. Burgard, and D. Fox, "Probabilistic robotics (intelligent robotics and autonomous agents series)," *Intelligent robotics and autonomous agents, The MIT . . .*, vol. 45, p. 52, 2005. [Online]. Available: http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Probabilistic+Robotics+(Intelligent+Robotics+and+Autonomous+Agents)#0

[28] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon, "Bundle Adjustment â A Modern Synthesis Vision Algorithms: Theory and Practice," in *Vision Algorithms: Theory and Practice*, 2000, vol. 1883, pp. 153–177. [Online]. Available: http://www.springerlink.com/content/plvcrq5bx753a2tn

[29] J.-S. Gutmann and B. Nebel, "Navigation mobiler Roboter mit Laserscans," in *Autonome Mobile Systeme (AMS97)*, 1997, pp. 36–47.

[30] J.-S. Gutmann and K. Konolige, "Incremental mapping of large cyclic environments," *Proceedings 1999 IEEE International Symposium on Computational Intelligence in Robotics and Automation. CIRA'99 (Cat. No.99EX375)*, 1999.

[31] K. Konolige, "Large-Scale Map-Making," in *The National Conference On Artificial Intelligence*, 2004, pp. 457–463. [Online]. Available: http://www.aaai.org/Papers/AAAI/2004/AAAI04-073.pdf

[32] M. Bosse, P. Newman, J. Leonard, M. Soika, W. Feiten, and S. Teller, "An Atlas framework for scalable mapping," *2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422)*, vol. 2, 2003.

[33] J. Folkesson and H. Christensen, "Graphical SLAM - a self-correcting map," *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004*, vol. 1, 2004.

[34] U. Frese, P. Larsson, and T. Duckett, "A multilevel relaxation algorithm for simultaneous localization and mapping," *IEEE Transactions on Robotics*, vol. 21, pp. 196–207, 2005.

[35] F. Dellaert and M. Kaess, "Square Root SAM: Simultaneous Localization and Mapping via Square Root Information Smoothing," pp. 1181–1203, 2006.

[36] U. Frese, "Treemap: An O(log n) algorithm for indoor simultaneous localization and mapping," *Autonomous Robots*, vol. 21, pp. 103–122, 2006.

[37] U. Frese and L. Schröder, "Closing a million-landmarks loop," in *IEEE International Conference on Intelligent Robots and Systems*, 2006, pp. 5032–5039.

[38] G. H. Golub and C. F. V. Loan, *Matrix Computations (Johns Hopkins Studies in Mathematical Sciences)(3rd Edition)*, 1996, vol. 208-209. [Online]. Available: http://www.amazon.ca/exec/obidos/redirect?tag=citeulike09-20&path=ASIN/0801854148

[39] M. Kaess, A. Ranganathan, and F. Dellaert, "Fast incremental square root information smoothing," in *IJCAI International Joint Conference on Artificial Intelligence*, 2007, pp. 2129–2134.

[40] T. A. Davis, J. R. Gilbert, S. I. Larimore, and E. G. Ng, "Algorithm 836: COLAMD, a column approximate minimum degree ordering algorithm," vol. 30, no. 3, pp. 377–380, Sept. 2004.

[41] D. Steedly, I. Essa, and F. Dellaert, "Spectral partitioning for structure from motion," *Proceedings Ninth IEEE International Conference on Computer Vision*, 2003.

[42] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," *Machine Learning*, vol. 1, no. 1, pp. 1–14, 2006. [Online]. Available: http://www.springerlink.com/index/y11g42n05q626127.pdf

[43] J. Shi and C. Tomasi, "Good features to track," *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition CVPR94*, vol. 94, no. June, pp. 593–600, 1994. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=323794

[44] J. Matas, O. Chum, M. Urban, and T. Pajdla, "Robust wide-baseline stereo from maximally stable extremal regions," *Image and Vision Computing*, vol. 22, no. 10, pp. 761–767, 2004. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S0262885604000435

[45] M. Agrawal, K. Konolige, and M. R. Blas, "Censure: Center surround extremas for realtime feature detection and matching," *Contract*, vol. 5305, pp. 102–115, 2008. [Online]. Available: http://www.springerlink.com/index/ngx06074r126x362.pdf

[46] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004. [Online]. Available: http://www.springerlink.com/openurl.asp?id=doi:10.1023/B:VISI.0000029664.99615.94

[47] H. Bay, T. Tuytelaars, L. Van Gool, and L. Van Gool, "Surf: Speeded up robust features," *Computer Vision – ECCV 2006*, vol. 3951, no. 3, pp. 404–417, 2006. [Online]. Available: http://eprints.pascal-network.org/archive/00002183/

[48] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "Brief : Binary robust independent elementary features," *Computer*, vol. 6314, no. 3, pp. 778–792, 2010. [Online]. Available: http://www.springerlink.com/index/H8H1824827036042.pdf

[49] B. D. Lucas and T. Kanade, *An Iterative Image Registration Technique with an Application to Stereo Vision (IJCAI)*, 1981, pp. 674–679.